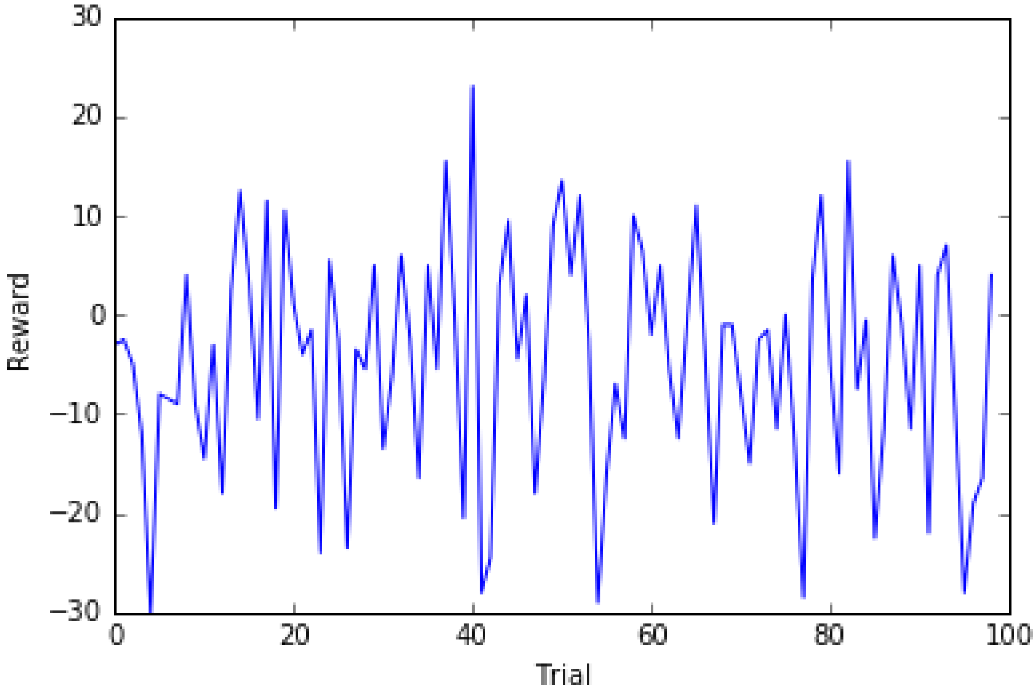
Smartcab Project

Implement a Basic Driving Agent

I modified the code to have the driving agent choose a random action. My observations are as follows:

- The smartcab regularly gets negative rewards;
- The smartcab sometimes makes it to the destination within the timeout limit (i.e. 100 steps over the deadline);
- For a particular run the average reward was -4.8 with 13 successes out of 100 trials. The plot of the reward for each trial is shown below.



Inform the Driving Agent

There are a number of pieces of information about the simulation that can be included in the agents state, they are as follows (with my chosen options in *italics*):

- *Waypoint*
- Inputs
    - *Light*
    - *Oncoming*
    - *Left*
    - Right
- Deadline

My reasoning is as follows:

- Waypoint is required so that the smartcab knows which direction to go;
- Light is necessary so the smartcab doesn't try to run red lights;
- Oncoming and Left are not essential as there are very few other cars on the road. Therefore information about the other cars could be ignored and the agent is unlikely to accrue too many penalties because of this. However, to allow the same agent to operate if there were more cars on the road I have chosen to include them in the state;
- Right is unnecessary as Light provides all the information needed to make a decision at a junction regardless of the value of Right;
- Deadline is unnecessary as it doesn't help the smartcab make decisions and adds to the 'curse of dimensionality'.

The total number of states is calculated by multiplying the number of possibilities for each variable together. Therefore the total number is 96 states (Waypoint[3] * Light[2] * Oncoming[4] * Left[4]).

Implement a Q-Learning Driving Agent

I implemented a Q-learning driving agent, I noticed the following:

- The smartcab now reaches its destination the majority of times;
- The smartcab reached its destination within the deadline the majority of times;
- The smartcab accrues few penalties and achieves much higher average rewards;
- The smartcab gets better at reaching its destination after a few trials.

This is all happening because the smartcab is learning from its previous actions. it is using the rewards given to it on previous actions to guide its future actions and hence gets better and better at getting to its destination and obeying the traffic rules.

Improve the Q-Learning Driving Agent

I improved the Q-learning driving agent by repeating the simulation a number of times looping over a selection of different values for the following variables (in all combinations):

- Learning rate (0, 0.05, 0.1, 0.5, 0.9, 0.95, 1)
- Discount factor (0, 0.05, 0.1, 0.5, 0.9, 0.95, 1)
- Exploration rate (0, 0.05, 0.1, 0.5, 0.9, 0.95, 1)
- Initial Q (0, 13)

I ran this three times and had the combinations that achieved the most successes (reached the destination the most times of the 100 trials) and that achieved the highest average reward printed out. The output of the three runs is as follows (multiple combinations achieved the same maximum number of successes):

Run 1

Maximum number of successes: 99.0

Learning rate: [0.05, 0.5, 0.9, 0.95, 1]

Discount factor: [0.95, 0.5, 0, 0, 0]

Exploration rate: [0.05, 0.05, 0, 0.05, 0]

Initial Q: [0, 0, 13, 13, 13]

Maximum average reward: 30.8282828283

Learning rate: [1]

Discount factor: [0.95]

Exploration rate: [0]

Initial Q: [13]

Run 2

Maximum number of successes: 99.0

Learning rate: [0.5, 0.95]

Discount factor: [0.1, 0.1]

Exploration rate: [0, 0.05]

Initial Q: [13, 0]

Maximum average reward: 29.7474747475

Learning rate: [0.05]

Discount factor: [0.95]

Exploration rate: [0]

Initial Q: [13]

Run 3

Maximum number of successes: 99.0

Learning rate: [0.5, 0.95]

Discount factor: [0.95, 0.1]

Exploration rate: [0, 0]

Initial Q: [13, 13]

Maximum average reward: 28.5303030303

Learning rate: [0.05]

Discount factor: [0.95]

Exploration rate: [0.1]

Initial Q: [13]

The results are different for each run due to the random nature of the simulation but some trends for the different settings are apparent:

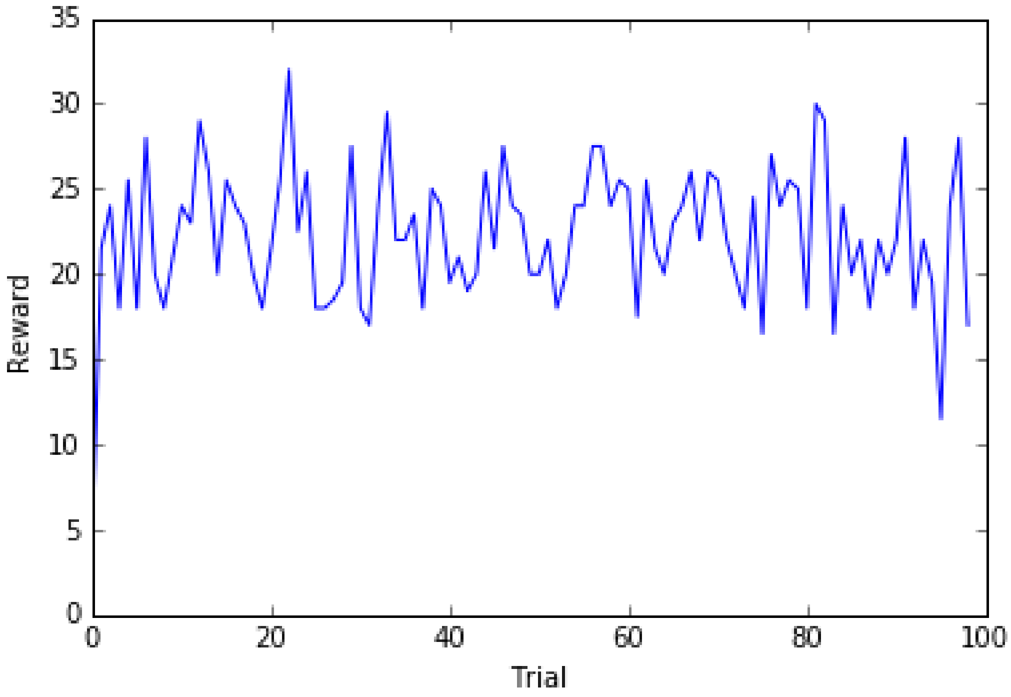
- In almost all cases the most successes and the highest average rewards are achieved when the initial Q is set to 13 rather than 0. This is because 13 is higher than the maximum possible reward and hence this means that the agent fully explores the state space and hence does not get stuck in a local optima.
- The exploration rate is low in all cases (less than or equal to 0.1). This is to be expected because if the exploration rate was high the agent would not be acting based on what it had learnt and instead would be acting almost randomly.

- The discount factor tends to be high to achieve the maximum average reward but to achieve the most successes there is no trend; sometimes a high discount factor comes out on top and other times a low discount factor wins.
- The optimum learning rate tends to be high (>0.5) to achieve the maximum number of successes but for the maximum average reward there is no obvious trend.

To assess how my agent behaves I chose the following settings:

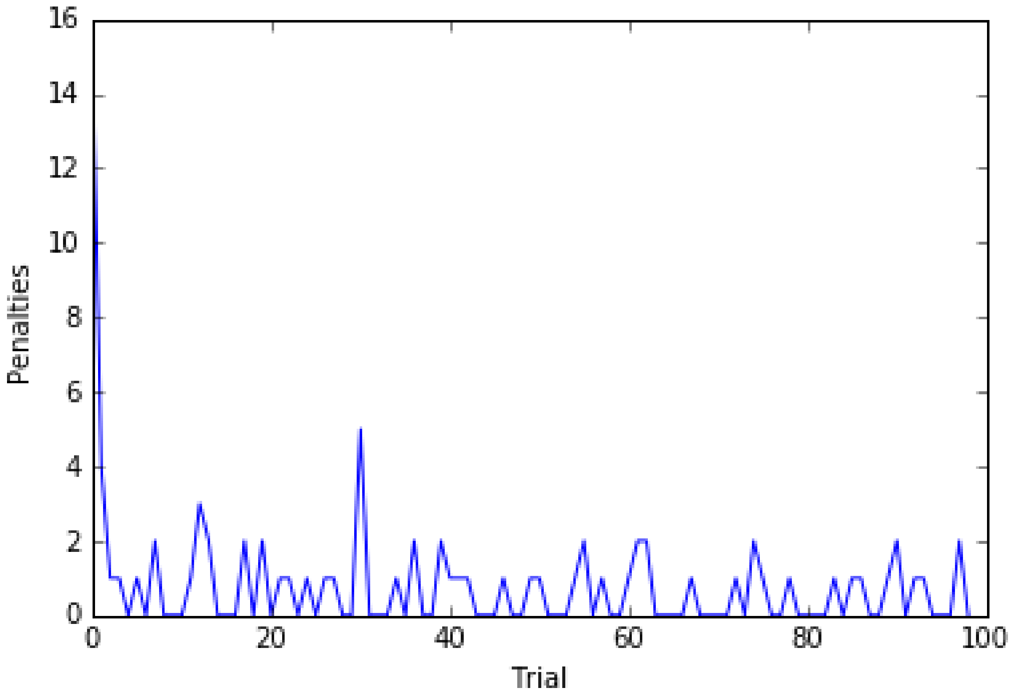
- Learning rate: 0.95
- Discount factor: 0.1
- Exploration rate: 0.05
- Initial Q: 13

For a particular run the average reward was 22.28 with 97 successes out of 100 trials. The plot of the reward for each trial is shown below.



Observing the smartcab shows that for the first one or two trials the smartcab moves seemingly randomly, accruing many penalties (this is the smartcab exploring the state space). However by the third trial the smartcab is moving directly to the destination accruing very few penalties. It is hard to say whether the smartcab is arriving at the destination in the shortest possible time but it is unlikely as the reward function does not reward getting to the destination quickly. To arrive in the quickest possible time the smartcab would need to know the timings of all of the traffic lights and attempt to take a route that meant hitting every light green so that it did not have to wait at any intersection. This information is not available to the smartcab so I expect that quicker routes are possible in many of the trials.

An optimal policy for this problem would be quite simple. In this scenario the driving agent just has to follow the planner and not accrue any penalties. Therefore, the policy would be 'If it is legal to do so follow the planner otherwise wait'. Some logic would be required to determine if the planned action was legal at the current time but this is relatively simple. To compare this optimal policy to the policy of my agent I have plotted the total number of penalties accrued for each trail below:



This plot shows that very quickly the agent stops accruing many penalties and in the second half of the trials it never accrues more than two in a single trail. The reason that it accrues any at all is likely to be because the exploration rate is non-zero and hence 1 in 20 actions are random. Therefore, this plot suggests that the learnt policy is close to optimal with the chosen settings.