P4 - Training a smartcab with Reinforcement Learning.

Julio Rodrigues (juliocezar.rodrigues@gmail.com) September 2016

1 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?

The agent sometimes does reach the destination by chance, but most of the time it drives around aimlessly and exceeds the deadline.

2 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?

Since the primary task here is to learn how to reach the destination without causing any accidents, the only sensor information used for this task are the light status (green or red), left/right/oncoming traffic and the next waypoint to be reached. These are the minimum necessary states for selecting the correct action to be taken next by any agent. $next\_waypoint$ tells where the agent should go next, while the surrounding traffic and the light status together allow one to reason upon which actions are legal to take (e.g.: $next\_waypoint$ is right, red light and no traffic coming forward from the left "tells" the agent that a right turn can be done.). At first, additionally to those, I also tried to use the current location, the final destination, and the deadline. This set of states increased the state space considerably and made learning very slow and inefficient. I also tried removing the deadline information from the status, but it did not improve the learning rate.

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?

In the setting described above, there are 384 possible states. In practice, however, even after running the simulation hundreds of times, only a handful different states are seen and learning still converges.

3 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?

Now, only after a few iterations, the agent learns to reach the destination consistently without breaking traffic laws. Over time the agent learns an optimal action-value function that gives the utility of taking a given action in a given state and register this value in a table. Later, when the agent finds itself in a state which it has visited before and therefore is on the table, it can simply select the action with the highest value for that state.

4 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?

There are several different initial alpha and epsilon values that performed very well in my grid search. I decided to set the initial values of both variables to $.5$.

Alpha is the learning rate which determines to what extent newly acquired information will override previously learned information and epsilon is a "randomness factor" when selecting a state, which determines the probability of a random action being taken, instead of taking a action based on a learned policy.

The initial epsilon value seems to be the most decisive factor when tuning the agent; very low values were very likely to worsen performance, specially with extremes values of alpha. The best combinations for these values are middle values for both. For low epsilon values, an alpha values around 0.5 perform the best.

Both values decay in decrements of 0.002 over time. Alpha is decremented until it reaches 0.001 and epsilon decays from 0.4 to 0.0. The reasoning behind letting these variables decay is the fact that the amount of randomness and the necessity for overriding old knowledge decrease over the lifetime of the agent, once it has seen a sufficient number of different states and accumulated knowledge about it so that it can choose an optimal policy.

A heatmap showing the different success rates for all combinations of alpha and epsilon is shown in figure 1:

The combination of a very low epsilon with a high or moderately high learning rate seems to be the cause of low success rates most of the time, while keeping both values balanced seems to deliver the best performance.

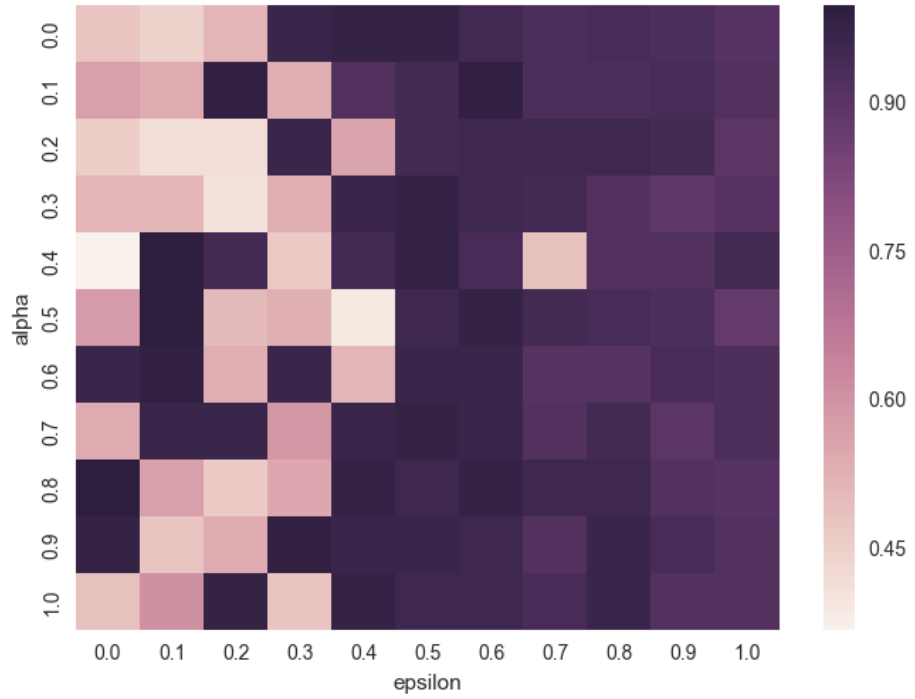

Figure 1: Heatmap for alpha-epsilon grid search

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?

Yes, it does. The success rate of the agent consistently reaches 98% or more, and it should be noted that most mistakes occur at the beginning of the training and only very occasionally at the end. And even the mistakes at the end usually don't involve the agent breaking traffic rules, but rather not following the direction set by the planner.

An optimal policy would be one which accurately follows the traffic rules and doesn't cause the agent to remain inactive when it doesn't have to or to follow a wrong direction which happens to return a higher reward.