Project 4: *Smartcab* - Self-Driving Cab

Robert Crowe

Robert@ourwebhome.com

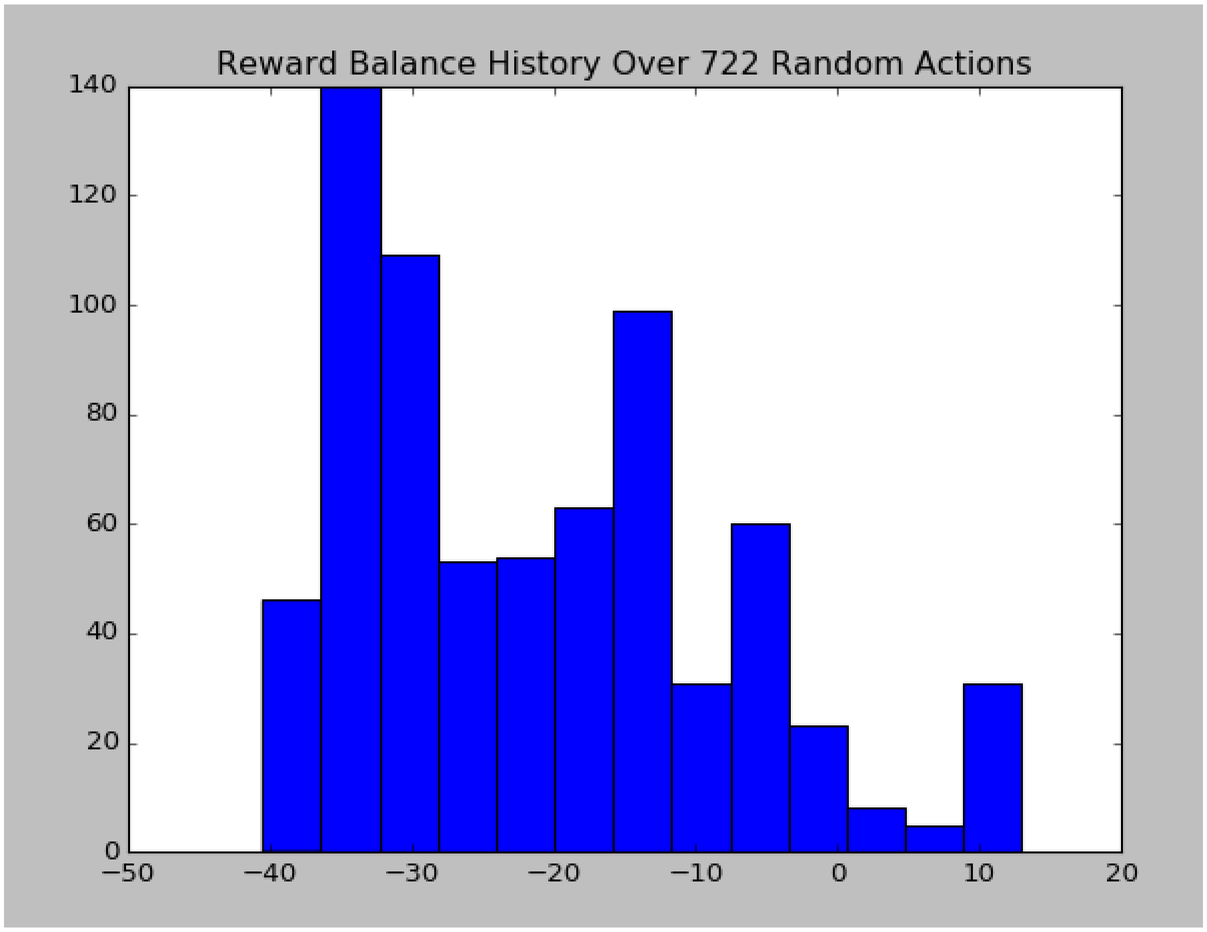
30 September 2016

Question 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?

Answer 1

As might be expected, when I randomly select actions the smartcab wanders around the grid. Since enforce_deadline is set to false it sometimes reaches the destination and receives that reward, although there is a hard time limit at -100 which often prevents it. Looking at a histogram of the accumulated rewards gives me an idea of the bias and distribution of the reward system values:



The rewards system appears to be skewed towards a negative balance, meaning that an agent that does not choose a good strategy is likely to end up with a negative rewards balance. This makes sense, since we want our rewards system to favor more efficient systems and encourage learning.

Question 2a - 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?

Answer 2a

The choice of states should be based on what we're trying to learn. It should also be designed for the minimum number of states in order to minimize the state space and training required ("Curse of Dimensionality") and be as easy to work with in the code as possible.

Since we're given the next waypoint, the main things that we need to learn are:

- The traffic laws
- The next best choice of action when choosing the next waypoint as the action would violate the traffic laws

That suggests that the only things that we need to encapsulate in the states are the current status at the intersection, and the next waypoint. The other dimension in the space are the actions, so that I have a state-action pair, since I need to learn a utility for each state-action pair and not simply each state. I've chosen to do that with a 2-D dictionary whose keys for the first dimension are pipe-separated concatenated strings for the status of each of the three directions at the intersection, plus the state of the light, the next waypoint, and the action for the second dimension.

We could also consider including the target length of the drive (the "deadline") as part of the state, since we are also given that information. However, we should always be careful when adding another dimension and increasing the size of the state space, since it will require more training to cover the space. Does the deadline give us valuable information that would make it worth increasing the size of the space?

Considering what we're trying to learn – the traffic laws and the next best choice – the deadline really doesn't give us any information that would inform those decisions. It doesn't relate to the traffic laws, and it doesn't tell us about the next best choice. Therefore, including it in the state would be a bad idea.

Question 2b - Inform the Driving Agent

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?

Answer 2b

I've chosen to represent the state-action pairs with a 2-D dictionary whose keys for the first dimension are pipe-separated concatenated strings for the status of each of the three directions at the intersection, plus the state of the light, the next waypoint, and the action for the second dimension. That gives me:

> 2 states for oncoming traffic in each direction
> ^ 3 directions
> X 2 states for the light
> X 3 directions for the next waypoint
>
> 2 ^ 3 * 2* 3 = 48 states

The learning space is actually the state-action pairs, since for each state different actions will have differ Q values. So the space is actually:

> 48 states X 4 actions = 192 state-action pairs

I could probably reduce the number of states by applying prior knowledge of the traffic laws, but since the goal is to learn the traffic laws that would be violating the design of the problem. For example, if I have a green light with a car on the left, I can only turn right or go forward, so the state could be right|forward as opposed to light-green|left-left|oncoming-None|right-None. That would result in:

> 1 + 2 + 3 = 6 legal actions
> X 3 directions for the next waypoint
> 6 X 3 = 18 states
> X 4 actions = 72 state-action pairs

Are 192 state-action pairs reasonable? Yes, given enough training. A smaller space would learn faster, but I need to stay within the parameters of the scenario.

Question 3 - Implement a Q-Learning Driving Agent

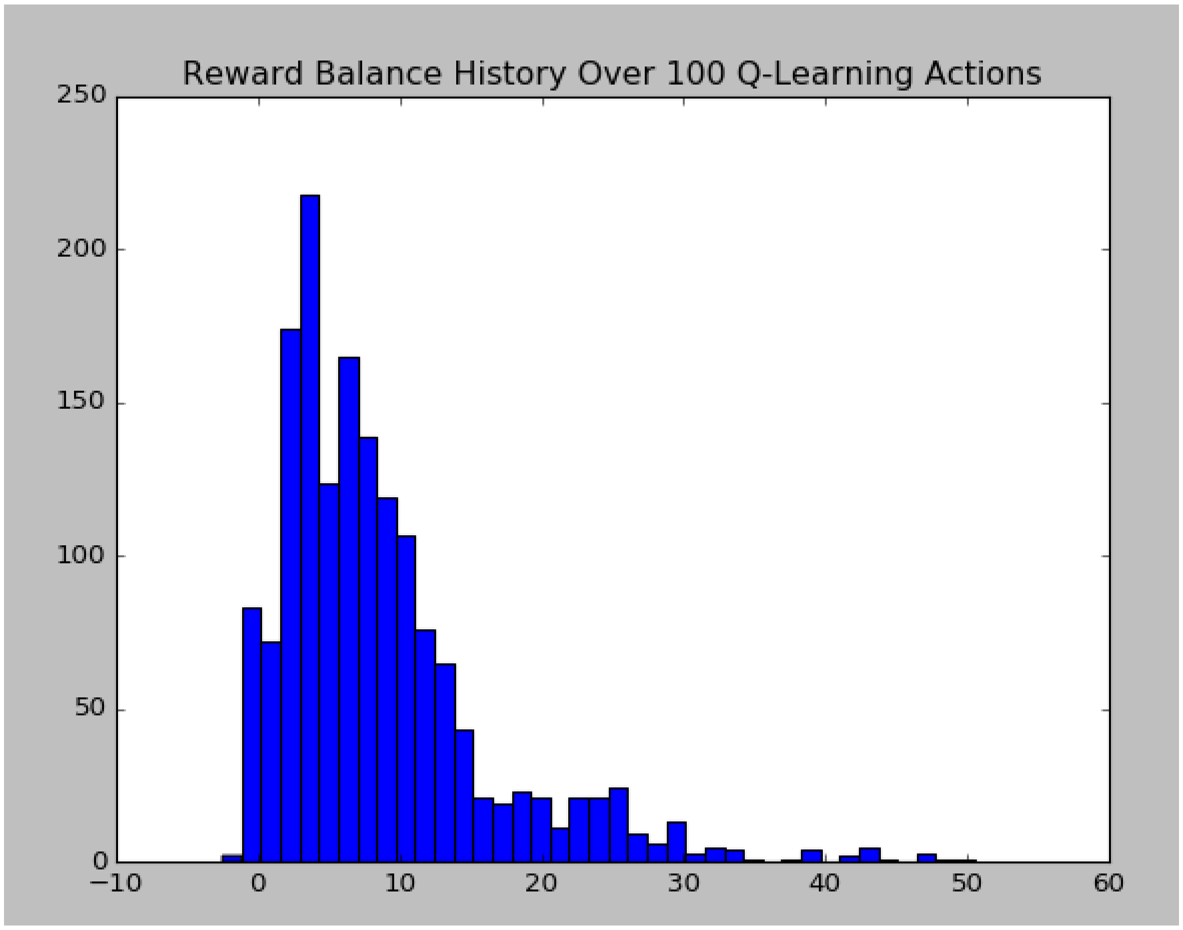
What changes do you notice in the agent's behavior when compared to the basic driving agent when random actions were always taken? Why is this behavior occurring?

Answer 3

The most striking change in behavior is that the agent consistently reached the destination with a positive reward.

When choosing actions randomly the agent simply wanders, without regard to traffic laws or movement towards the destination. At the beginning of training with Q-learning the agent behaves in essentially the same way, but quickly starts to display more constructive behavior as it begins to learn. As learning increases it makes fewer traffic mistakes, and tends to move towards the destination with greater consistency.

In a typical test with 100 trials, with enforce_deadline=True, the agent reaches the destination in 85% of the trials, with a positive rewards distribution:



My initial values for the hyperparameters were:

self.alpha = 0.1
self.gamma = 0.9
self.epsilon = 0.1

Question 4a - 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?

Answer 4a

It seemed to make sense to do a grid search over the alpha, gamma, and epsilon hyperparameters, measured by the success rate of reaching the destination and the number of penalties incurred along the way. Intuitively, we want the cab to reach the destination in a reasonable amount of time, and we don't want to break traffic laws to do it. (Some observers may comment that this is often unlike actual human-driven cabs) So we want to maximize the success rate of reaching the destination in the allotted time, and minimize the penalties along the way. However, reaching the destination is a somewhat stochastic measure, since each different environment is initialized randomly.

With static (non-decaying) values, the results were that optimal values are in the neighborhood of:

Alpha = 0.3
Gamma = 0.1
Epsilon = 0.1

which successfully reaches the destination in approximately 95% of the trials, and incurs penalties in approximately 9% of actions.

Another strategy is to decay alpha as 1/t in order to encourage convergence. Using this approach, the optimal values for successfully reaching the destination seem to be in the neighborhood of:

Starting Alpha = 0.3
Gamma = 0.5
Epsilon = 0.1

which achieved a success rate of approximately 99%.

Optimal values form minimizing penalties seem to be in the neighborhood of:

Starting Alpha = 0.1
Gamma = 0.1
Epsilon = 0.1

which achieved a penalty rate of approximately 8%.

Another strategy is during exploration to favor, when choosing an action randomly for a given state, any actions that have not been tried yet. The random choice is then between only those actions that have not yet been tried for this state.

This is similar to increasing epsilon, but specifically focused towards reducing the sparsity of the matrix. It also just makes sense on an intuitive level to favor trying things that you haven't tried before. Using this approach, the optimal values for successfully reaching the destination seem to be in the neighborhood of:

Starting Alpha = 0.3
Gamma = 0.2
Epsilon = 0.1

which achieved a success rate of approximately 99%.

Optimal values form minimizing penalties seem to be in the neighborhood of:

Starting Alpha = 0.1
Gamma = 0.1
Epsilon = 0.1

which achieved a penalty rate of approximately 7.7%.

Another strategy is to severely penalize any (state,action) pairs that result in a penalty by setting their Q-value to a large negative value. This avoids repeating the same action and getting the same negative result, sometimes referred to as the "if it hurts when you do that, stop doing that" principle. Using this approach, the optimal values for successfully reaching the destination seem to be in the neighborhood of:

Starting Alpha = 0.1
Gamma = 0.3
Epsilon = 0.1

which achieved a success rate of approximately 99%.

Optimal values form minimizing penalties seem to be in the neighborhood of:

Starting Alpha = 0.1
Gamma = 0.1
Epsilon = 0.1

which achieved a penalty rate of approximately 9%.

Counter-intuitively, increasing penalties for bad choices does not seem to significantly decrease the number of penalties incurred.

Question 4b - Improve the Q-Learning Driving Agent

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?

Answer 4b

It is approaching an optimal policy, but in my view the level of penalties is still too high. Reaching the destination on time in 99% of journeys is probably acceptable, but 8% penalties seems too high - higher than your average cab driver would do.

Penalties were mostly due to exploration of new state-action pairs. As part of training the model needs to see negative rewards in order to learn what state-actions violate traffic laws. Roughly 88.47% of penalties occurred as a result of exploration, and probably a high percentage of the remainder occurred on the second visit to a state-action pair.

To test whether increased training would improve results, I tried setting n_trials=200. However those results also show no significant improvement in either the success in reaching the destination or the number of penalties along the way.

Optimal policy would be to never violate traffic laws, meaning that we would never incur penalties. In practice a self-driving cab would need to be trained sufficiently so that it could be expected to rarely if ever violate traffic laws, as that could endanger the passengers and others on the road. Secondarily, an optimal policy would choose the shortest route to the destination, again without violating traffic laws. Both of these behaviors – driving within the law and choosing the shortest route – are what we typically expect of human cab drivers. It is reasonable to expect the same from smartcabs.