Assignment Report

Project 4: Train a Smart cab to Drive

Udacity Machine Learning Nanodegree Program
Authored by: Fedor Sulaev
Date: 03/08/2016

1. Implement a basic driving agent

The basic driving agent produces random action on each step. The agent is able to "find" the target location eventually due to the limited size of the map and finite number of choices, but the reward value for this type of agent tends to be low because the agent ignores all rules and state information.

2. Identify and update state

The following state variables represent the driver's state:

- Light – intersection traffic light (red or green).
- Oncoming – traffic from the opposite direction
- Left – traffic from the left side
- Right – traffic from the right side
- Next waypoint – direction to the next waypoint on the route to the destination (forward, left or right)

In order to minimize the number of redundant dimensions only relative directions used in the driver's state instead of the variables representing agent or goal positions on the board in [x, y] coordinates.

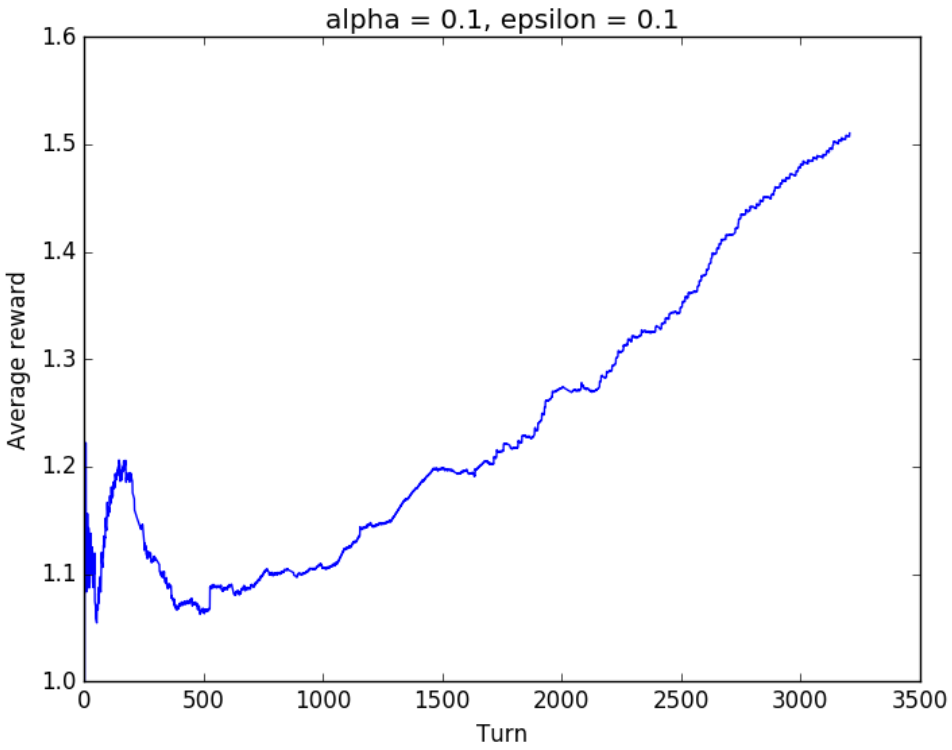
3. Implement Q-Learning

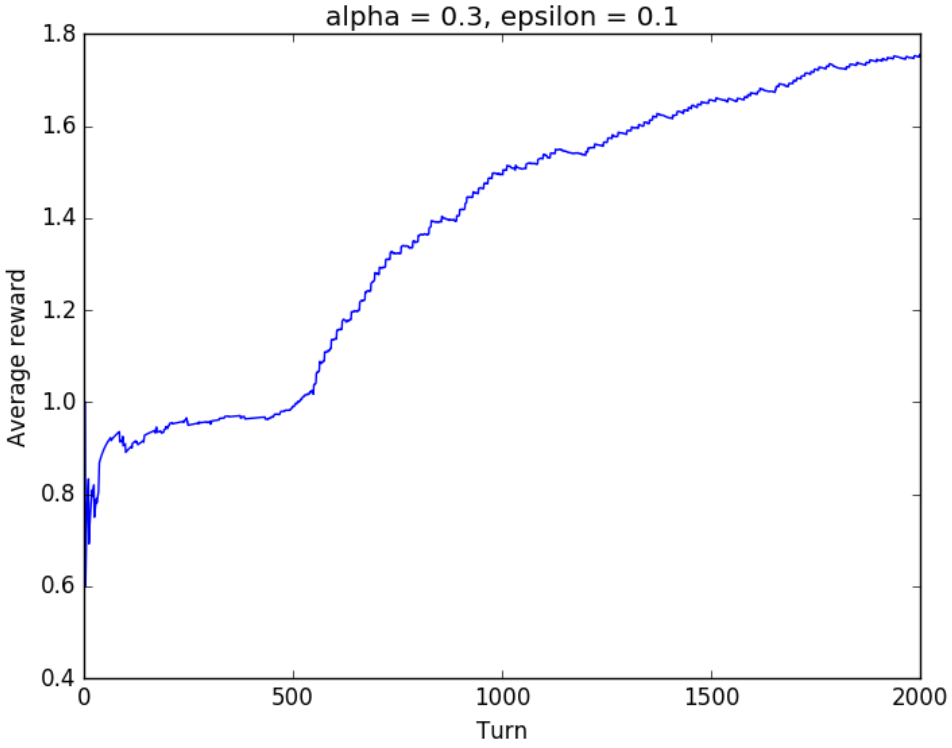
The agent uses epsilon-greedy Q-Learning algorithm, which means that ϵ of the time it selects random action and $1 - \epsilon$ of the time it uses action with the highest estimated value. If the current state of the agent wasn't observed before, the agent selects random action. This strategy allows the agent to explore new state-action combinations that would have been ignored otherwise.

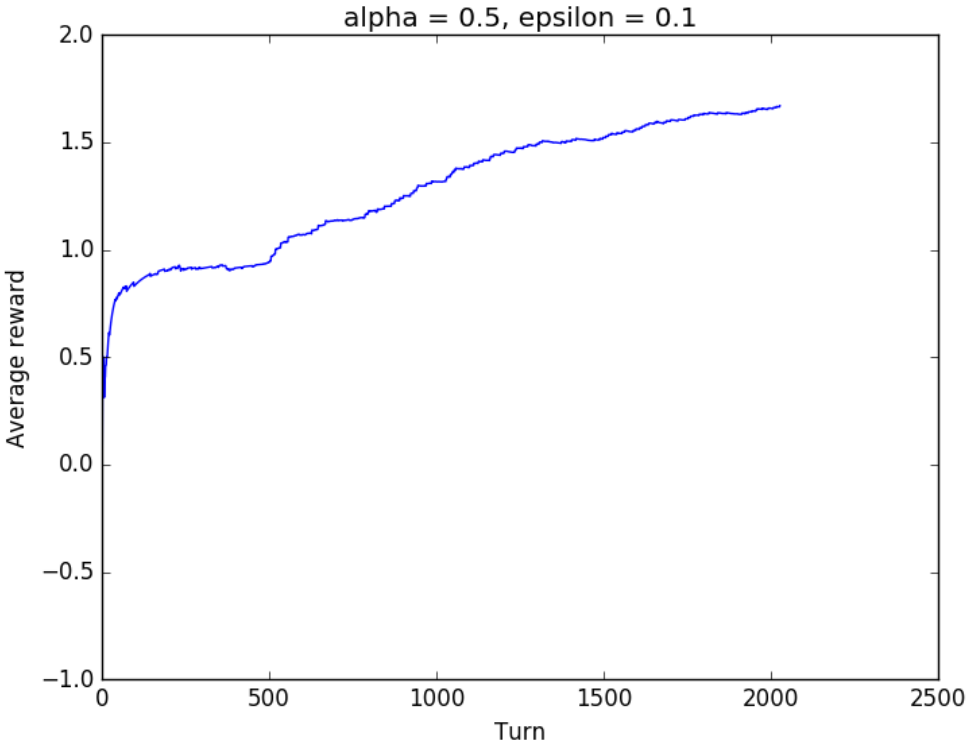
The advanced agent at the beginning of the training tends to get stuck in suboptimal greedy action choices but after some exploration it finds more valuable state-action combinations and the number of turns required for the agent to reach its goal becomes much smaller than for the basic agent that uses random choices.

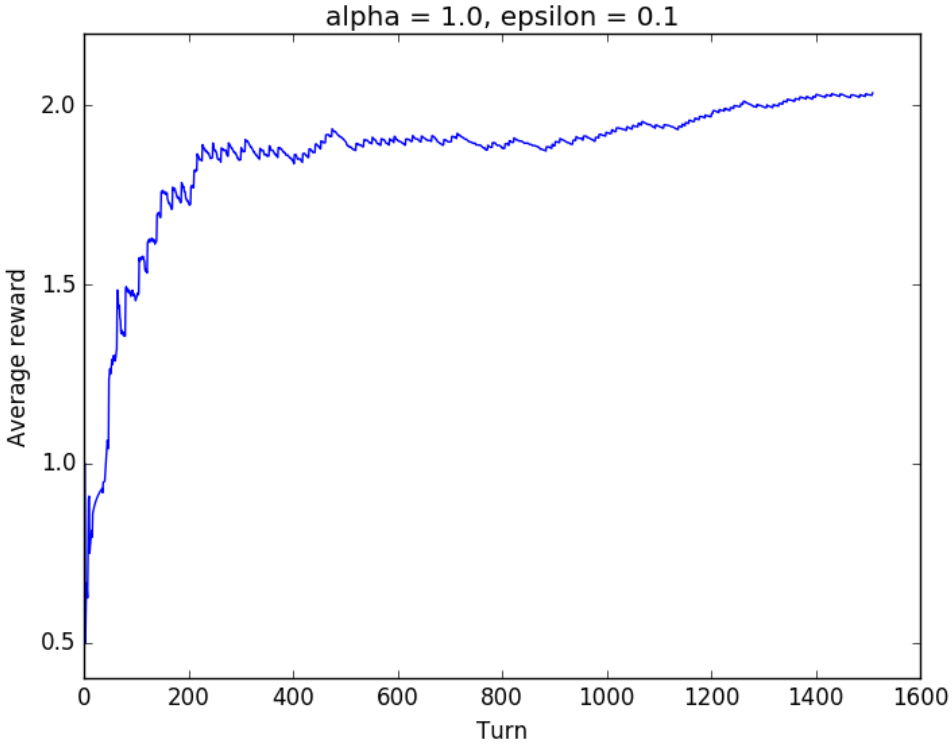
4. Enhance the driving agent

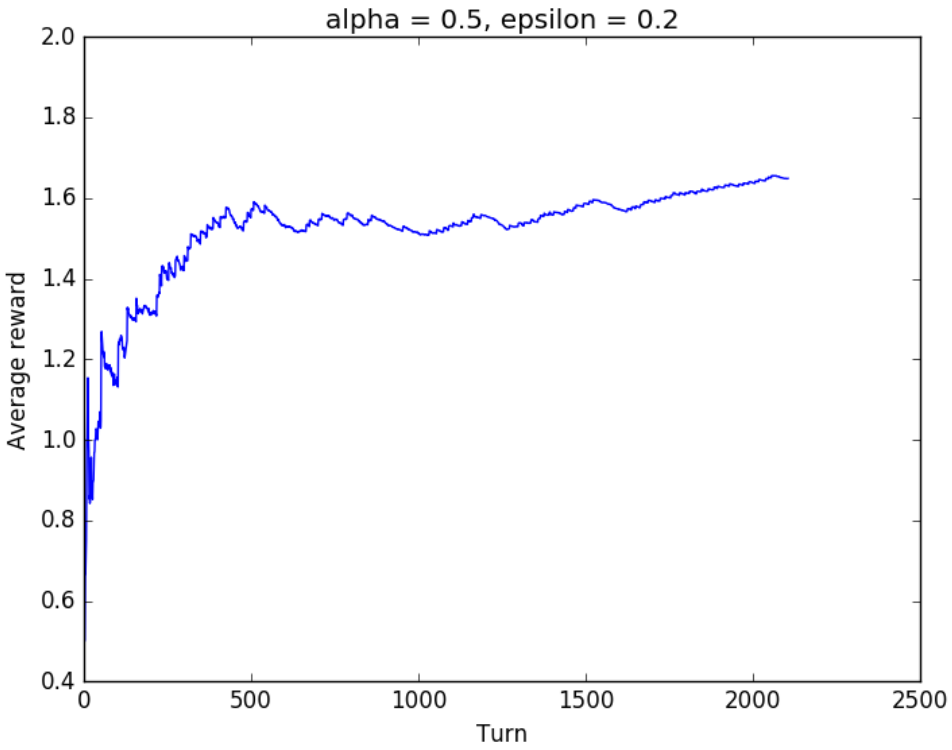
Different values were tested for learning rate (α) and ϵ parameters, the graphs below show correlation between average reward and number of turns as well as the total number of turns that the agent took in order to pass 100 trials.

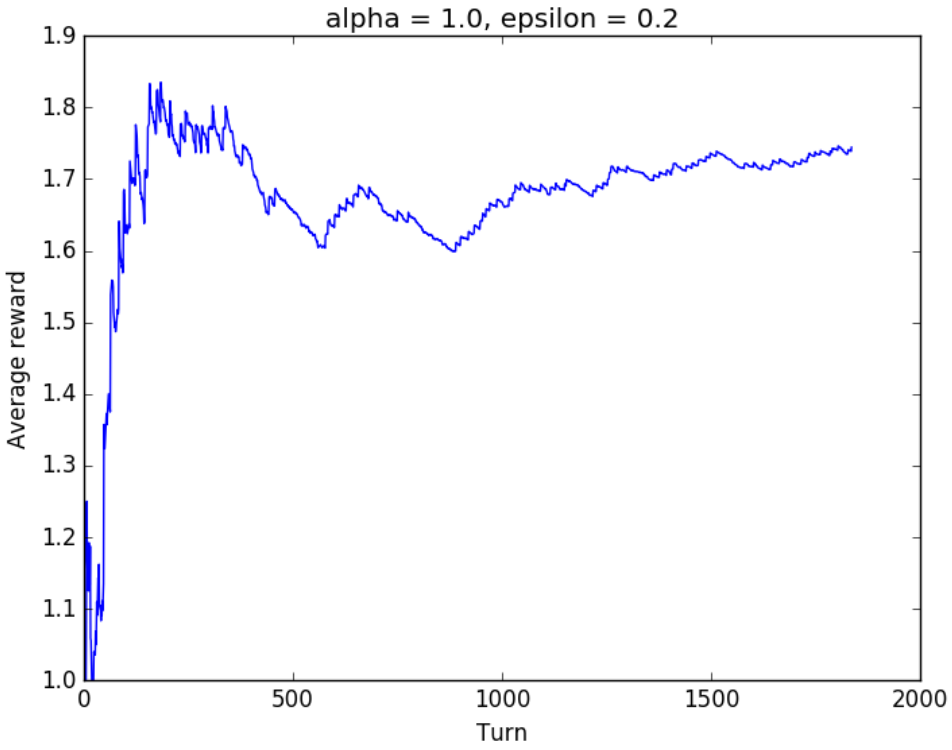

alpha = 0.1, epsilon = 0.1
Average reward
Turn
1.0
1.1
1.2
1.3
1.4
1.5
1.6
0
500
1000
1500
2000
2500
3000
3500


alpha = 0.3, epsilon = 0.1
Average reward
Turn
0.4
0.6
0.8
1.0
1.2
1.4
1.6
1.8
0
500
1000
1500
2000


alpha = 0.5, epsilon = 0.1
Average reward
Turn
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
0
500
1000
1500
2000
2500


alpha = 1.0, epsilon = 0.1
Average reward
Turn
2.0
1.5
1.0
0.5
0
200
400
600
800
1000
1200
1400
1600


alpha = 0.5, epsilon = 0.2
Average reward
Turn
2.0
1.8
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0
500
1000
1500
2000
2500


alpha = 1.0, epsilon = 0.2
Average reward
Turn
1.9
1.8
1.7
1.6
1.5
1.4
1.3
1.2
1.1
1.0
0
500
1000
1500
2000