Project 4: Reinforcement Learning

Train a Smartcab to Drive

Project Overview

In this project you will apply reinforcement learning techniques for a self-driving agent in a simplified world to aid it in effectively reaching its destinations in the allotted time. You will first investigate the environment the agent operates in by constructing a very basic driving implementation. Once your agent is successful at operating within the environment, you will then identify each possible state the agent can be in when considering such things as traffic lights and oncoming traffic at each intersection. With states identified, you will then implement a Q-Learning algorithm for the self-driving agent to guide the agent towards its destination within the allotted time. Finally, you will improve upon the Q-Learning algorithm to find the best configuration of learning and exploration factors to ensure the self-driving agent is reaching its destinations with consistently positive results.

Description

In the not-so-distant future, taxicab companies across the United States no longer employ human drivers to operate their fleet of vehicles. Instead, the taxicabs are operated by self-driving agents? known as **smartcabs**? to transport people from one location to another within the cities those companies operate. In major metropolitan areas, such as Chicago, New York City, and San Francisco, an increasing number of people have come to rely on **smartcabs** to get to where they need to go as safely and efficiently as possible. Although **smartcabs** have become the transport of choice, concerns have arisen that a self-driving agent might not be as safe or efficient as human drivers, particularly when considering city traffic lights and other vehicles. To alleviate these concerns, your task as an employee for a national taxicab company is to use reinforcement learning techniques to construct a demonstration of a **smartcab** operating in real-time to prove that both safety and efficiency can be achieved.

Definitions

Environment

The **smartcab** operates in an ideal, grid-like city (similar to New York City), with roads going in the North-South and East-West directions. Other vehicles will certainly be present on the road, but there will be no pedestrians to be concerned with. At each intersection is a traffic light that either allows traffic in the North-South direction or the East-West direction. U.S. Right-of-Way rules apply:

- On a green light, a left turn is permitted if there is no oncoming traffic making a right turn or coming straight through the intersection.
- On a red light, a right turn is permitted if no oncoming traffic is approaching from your left through the intersection. To understand how to correctly yield to oncoming traffic when turning left, you may refer to this official drivers? education video, or this passionate exposition.

Inputs and Outputs

Assume that the **smartcab** is assigned a route plan based on the passengers? starting location and destination. The route is split at each intersection into waypoints, and you may assume that the **smartcab**, at any instant, is at some intersection in the world. Therefore, the next waypoint to the destination, assuming the destination has not already been reached, is one intersection away in one direction (North, South, East, or West). The **smartcab** has only an egocentric view of the intersection it is at: It can determine the state of the traffic light for its direction of movement, and whether there is a vehicle at the intersection for each of the oncoming directions. For each action, the **smartcab** may either idle at the intersection, or drive to the next intersection to the left, right, or ahead of it. Finally, each trip has a time to reach the destination which decreases for each action taken (the passengers want to get there quickly). If the allotted time becomes zero before reaching the destination, the trip has failed.

Rewards and Goal

The **smartcab** receives a reward for each successfully completed trip, and also receives a smaller reward for each action it executes successfully that obeys traffic rules.
The **smartcab** receives a small penalty for any incorrect action, and a larger penalty for any action that violates traffic rules or causes an accident with another vehicle. Based on the rewards and penalties the **smartcab** receives, the self-driving agent implementation

should learn an optimal policy for driving on the city roads while obeying traffic rules, avoiding accidents, and reaching passengers? destinations in the allotted time.

Tasks

Project Report

You will be required to submit a project report along with your modified agent code as part of your submission. As you complete the tasks below, include thorough, detailed answers to each question *provided in italics*.

Implement a Basic Driving Agent

To begin, your only task is to get the **smartcab** to move around in the environment. At this point, you will not be concerned with any sort of optimal driving policy. Note that the driving agent is given the following information at each intersection:

- The next waypoint location relative to its current location and heading.
- The state of the traffic light at the intersection and the presence of oncoming vehicles from other directions.
- The current time left from the allotted deadline.

To complete this task, simply have your driving agent choose a random action from the set of possible actions (`None`, `'forward'`, `'left'`, `'right'`) at each intersection, disregarding the input information above. Set the simulation deadline enforcement, `enforce_deadline` to `False` and observe how it performs.

QUESTION: *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: *The smartcab just moves in random directions. It does not mind traffic or where the destination is located. It does get to its destination occasionally even within the deadline. But sometimes it does not reach its destination even when the deadline counted down to minus 100 where the trial will be aborted. It happens all by chance. The smaller the distance between the car's start position and its destination, the higher the chances it will reach its destination.*

Inform the Driving Agent

Now that your driving agent is capable of moving around in the environment, your next task is to identify a set of states that are appropriate for modeling the **smartcab** and environment. The main source of state variables are the current inputs at the intersection, but not all may require representation. You may choose to explicitly define states, or use some combination of inputs as an implicit state. At each time step, process the inputs and update the agent's current state using the `self.state` variable. Continue with the simulation deadline enforcement `enforce_deadline` being set to `False`, and observe how your driving agent now reports the change in state as the simulation progresses.

QUESTION: *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: *All the information we have, is the next waypoint and the current state of traffic. The next waypoint can be "right", "forward" or "left". The states of traffic depend on the lights ("green" or "red"), oncoming traffic ("None", "right", "forward" or "left") and the traffic coming from the left side ("None", "right", "forward" or "left"). Because of the USA right of way rules, we can exclude the traffic which comes from the right side. It would just lead to a larger number of states and had no influence of how the cab should act in a situation. We also exclude the deadline. The deadline is an integer and would multiply the number of states by its maximum initial value. Working with so many states would take a lot more time to train the cab which we do not have. Another point to keep in mind is the ethical issue to include the deadline. Including the deadline would mean to violate the traffic rules occasionally and risk people's lives.*
*In total we have 2 * 4 * 4 = 32 traffic states. Since each traffic state may occur together with each waypoint, we need to take all possible combinations in account and get a total number of 96 states.*

OPTIONAL: *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: *We have 96 states in total. In each state, the car has four options on how to act. These are enough states for the Q-Learning algorithm to know how to act in every situation. But it could be to many states to train the cab in this short time, since the cab should try each possible action in each possible state at least once to find the best action for each state.*

Implement a Q-Learning Driving Agent

With your driving agent being capable of interpreting the input information and having a mapping of environmental states, your next task is to implement the Q-Learning algorithm for your driving agent to choose the *best* action at each time step, based on the Q-values for the current state and action. Each action taken by the **smartcab** will produce a reward which depends on the state of the environment. The Q-Learning driving agent will need to consider these rewards when updating the Q-values. Once implemented, set the simulation deadline enforcement `enforce_deadline` to `True`. Run the simulation and observe how the **smartcab** moves about the environment in each trial. The formulas for updating Q-values can be found in this video.

QUESTION: *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: *How the cab acts depends a lot on its first experiences. If the cab makes all the correct choices at the beginning, it builds up q-values which let the cab repeat this actions and lead it always to its goal. If the cab takes some good and some bad actions at the beginning, it will reach its goal way less often or might never get to its goal at all. Sometimes it moves in a circle just taking right turns. I guess this behavior occurs because the cab does not try different actions anymore as soon as it found a q-value which is greater than zero for a state. The reward for a correct move is 2 and a wrong move gets a reward of minus 1 or minus 0.5. This means, if a bad action could bring the cab in a state from which it could make a correct action, it would get a positive q-value for that bad action and would chose it every time.*

Improve the Q-Learning Driving Agent

Your final task for this project is to enhance your driving agent so that, after sufficient training, the **smartcab** is able to reach the destination within the allotted time safely and efficiently. Parameters in the Q-Learning algorithm, such as the learning rate (`alpha`), the discount factor (`gamma`) and the exploration rate (`epsilon`) all contribute to the driving agent's ability to learn the best action for each state. To improve on the success of your **smartcab**:

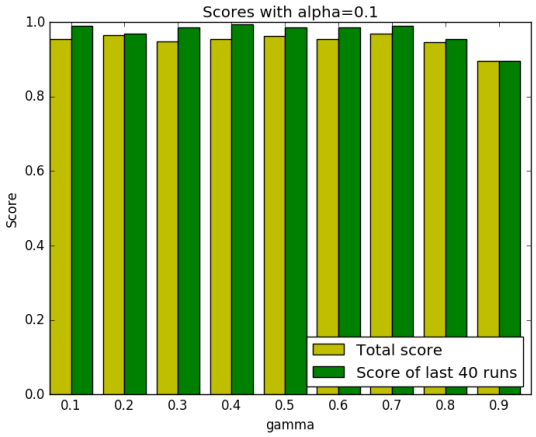
- Set the number of trials, `n_trials`, in the simulation to 100.
- Run the simulation with the deadline enforcement `enforce_deadline` set to `True` (you will need to reduce the update delay `update_delay` and set the `display` to `False`).
- Observe the driving agent's learning and **smartcab's** success rate, particularly during the later trials.

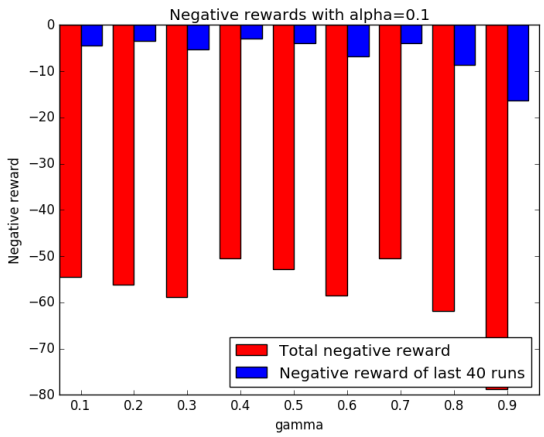
- Adjust one or several of the above parameters and iterate this process.

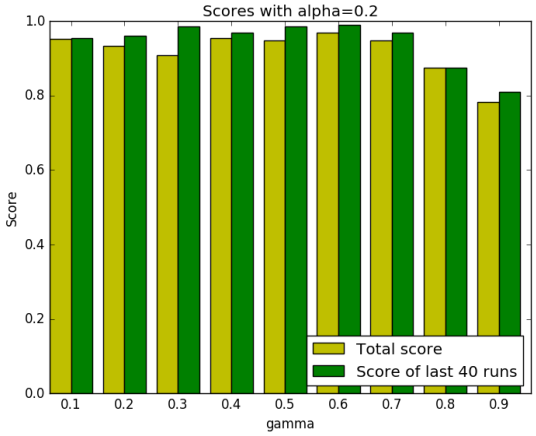
This task is complete once you have arrived at what you determine is the best combination of parameters required for your driving agent to learn successfully.

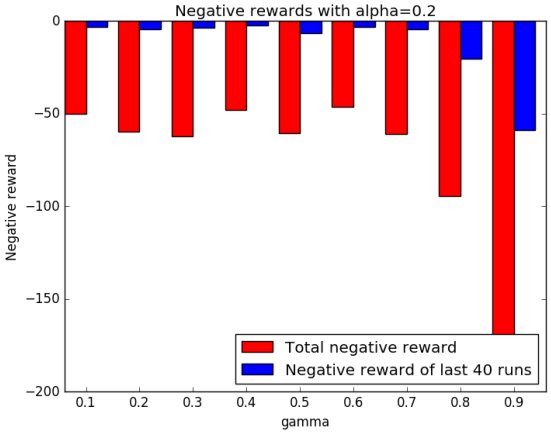
QUESTION: *Report the different values for the parameters tuned in your basic implementation of Q-Learning. For which set of parameters does the agent perform best? How well does the final driving agent perform?*

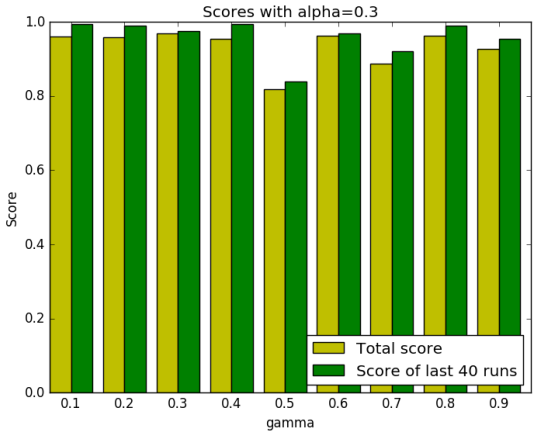
ANSWER: *For epsilon, I chose a start value of 0.2 and reduced it over time becoming zero after 50 trials. I also chose to reduce alpha step by step until it becomes zero after 100 trials. In order to find the best values for alpha and gamma, I decided to do all the runs for these values from 0.1 to 0.9 for both variables. Every value pair ran five times and the average values are displayed in the charts below this text. According to these charts, the best value for alpha is 0.6 and the best value for gamma is 0.3. The final driving agent performs well when it comes to reaching the goal. In this report are 6 charts which shows when the driving agent reached its goal and how many penalties it got. In these charts, it got a scores between 0.94 and 0.99. But it still makes mistakes and violates the traffic rules.*

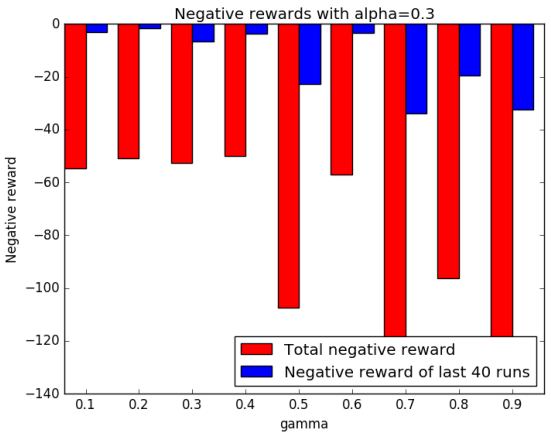

Scores with alpha=0.1
Score
gamma
Total score
Score of last 40 runs

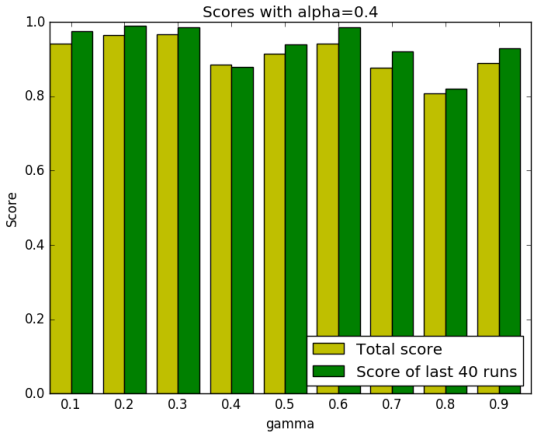

Negative rewards with alpha=0.1
Negative reward
gamma
Total negative reward
Negative reward of last 40 runs

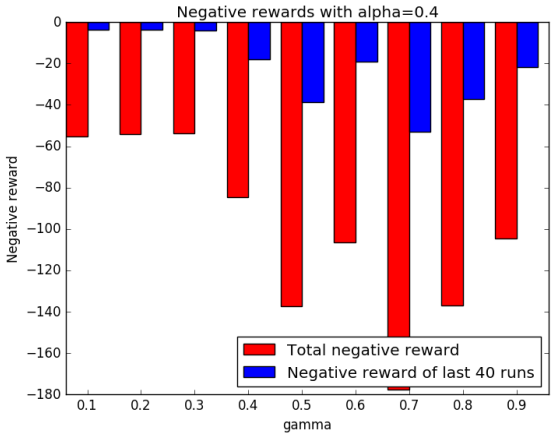

Scores with alpha=0.2
Score
gamma
Total score
Score of last 40 runs

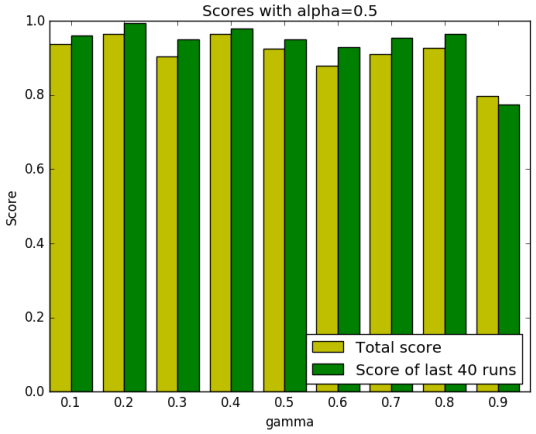

Negative rewards with alpha=0.2
Negative reward
gamma
Total negative reward
Negative reward of last 40 runs

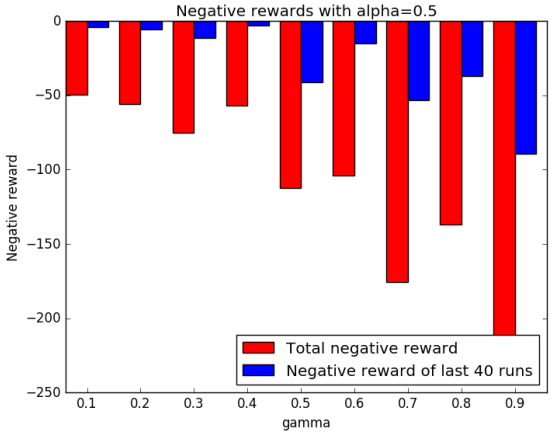

Scores with alpha=0.3
Score
gamma
Total score
Score of last 40 runs

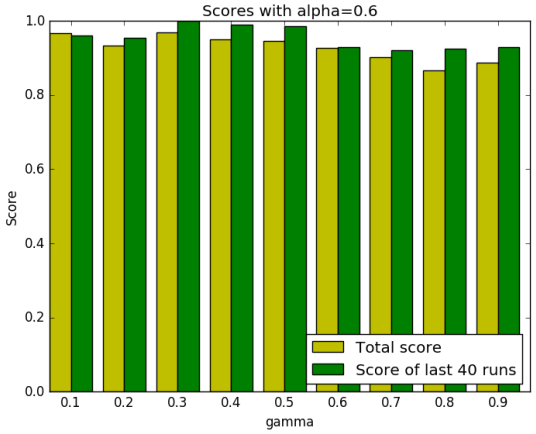

Negative rewards with alpha=0.3
Negative reward
gamma
Total negative reward
Negative reward of last 40 runs

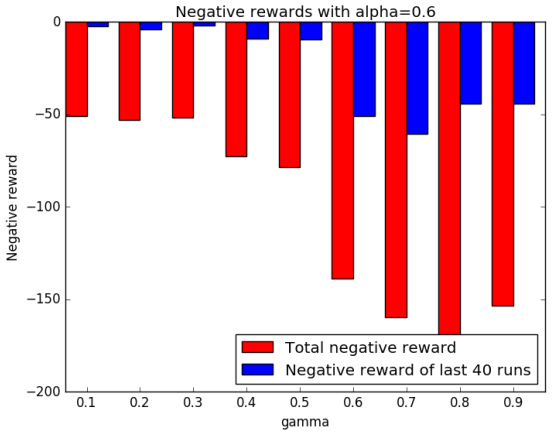

Scores with alpha=0.4
Score
gamma
Total score
Score of last 40 runs

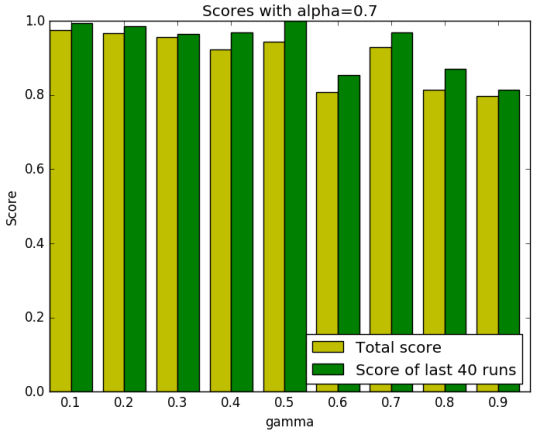

Negative rewards with alpha=0.4
Negative reward
gamma
Total negative reward
Negative reward of last 40 runs

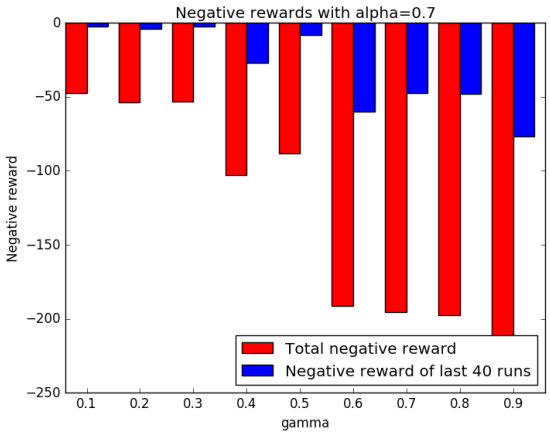

Scores with alpha=0.5
Score
gamma
Total score
Score of last 40 runs

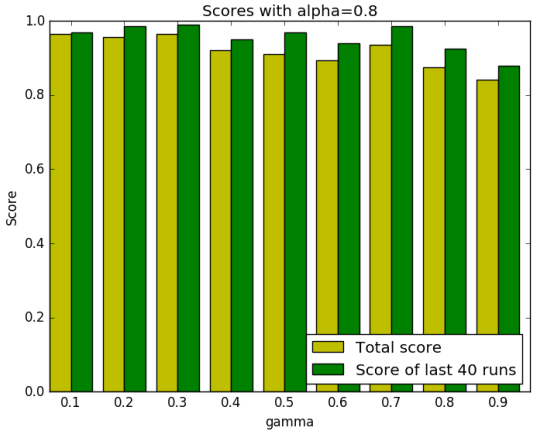

Negative rewards with alpha=0.5
Negative reward
gamma
Total negative reward
Negative reward of last 40 runs

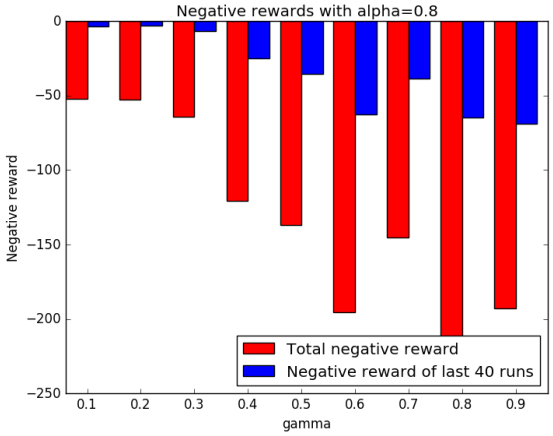

Scores with alpha=0.6
Score
gamma
Total score
Score of last 40 runs

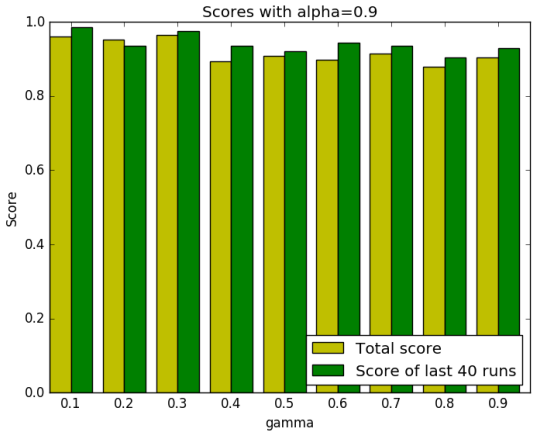

Negative rewards with alpha=0.6
Negative reward
gamma
Total negative reward
Negative reward of last 40 runs

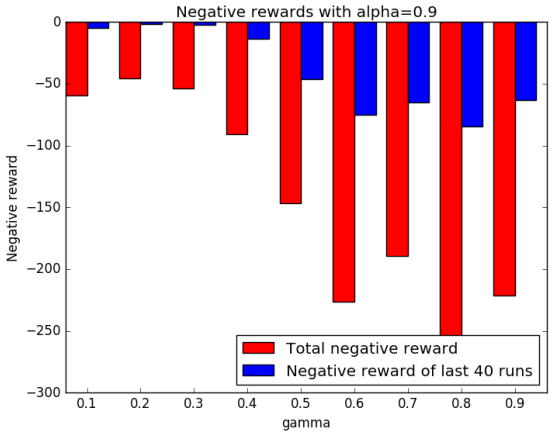

Scores with alpha=0.7
Score
gamma
Total score
Score of last 40 runs


Negative rewards with alpha=0.7
Negative reward
gamma
Total negative reward
Negative reward of last 40 runs

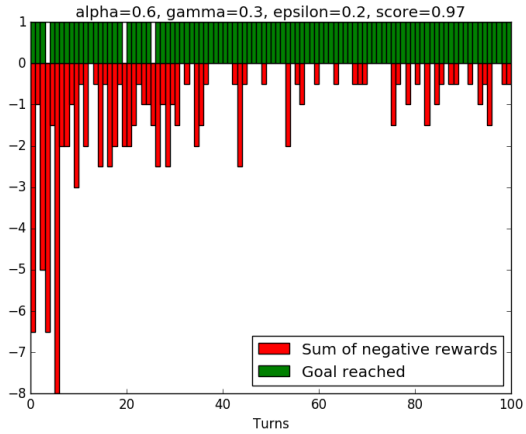

Scores with alpha=0.8
Score
gamma
Total score
Score of last 40 runs

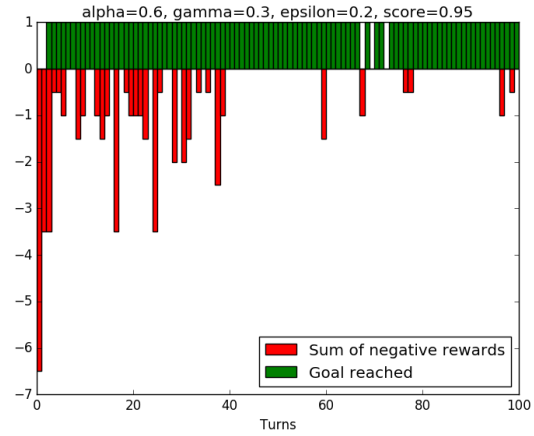

Negative rewards with alpha=0.8
Negative reward
gamma
Total negative reward
Negative reward of last 40 runs

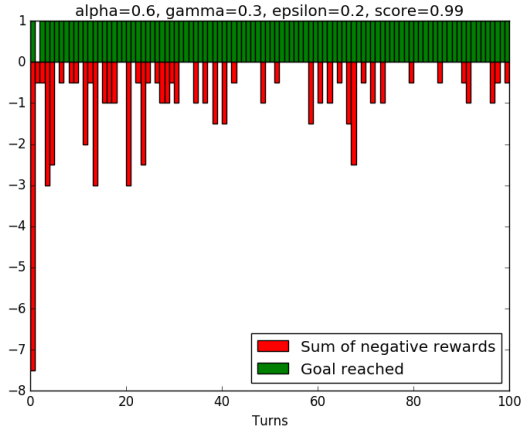

Scores with alpha=0.9
Score
gamma
Total score
Score of last 40 runs

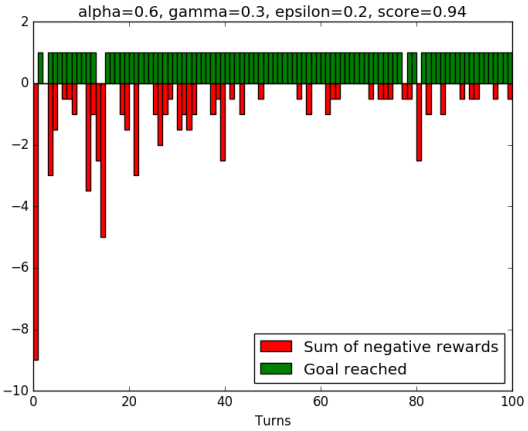

Negative rewards with alpha=0.9
Negative reward
gamma
Total negative reward
Negative reward of last 40 runs

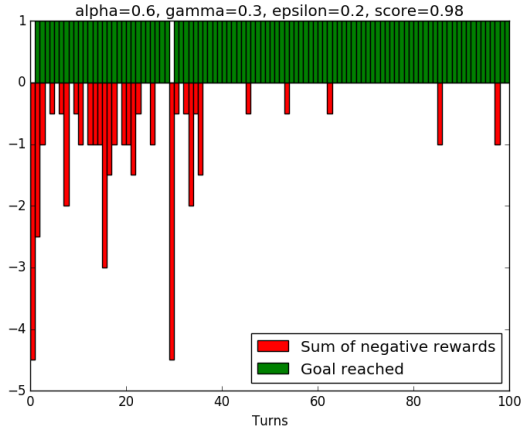
These are six charts of six different runs with the optimal parameter values. It is clear to see that the cab usually reaches its goal, but still makes mistakes and violates the traffic regulations.

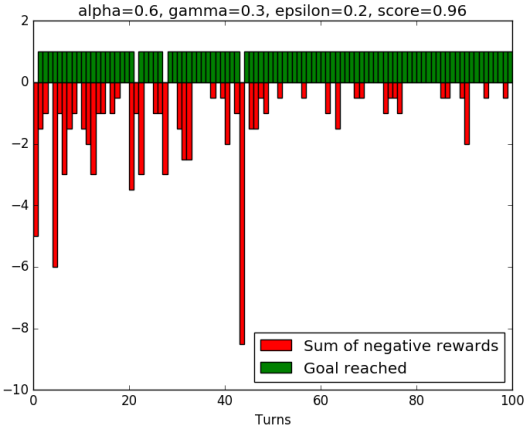












QUESTION: *Does your agent get close to finding an optimal policy, i.e. reach the destination in the minimum possible time, and not incur any penalties? How would you describe an optimal policy for this problem?*

ANSWER: *In my opinion, the optimal policy would be to follow the next waypoint and comply with the traffic regulations in all states. In other words: "If the traffic regulations allow to drive to the next waypoint, drive to the next waypoint. If driving to the next waypoint would violate the traffic regulations, do nothing."*
The driving agent has still a lot of undiscovered states and will take unexpected actions when it comes to new situations. It needs a lot more training and discover all possible states in order to operate alone on the street. The following plots are from the same runs as the plots from the last questions. They show how many times the driving agent took the optimal action according to the optimal policy within the last 40 iterations. In these six runs, it scores between 0.91 and 0.98. It goes in the right way, but it is not ready for the streets yet. With sufficient training, it would take all the actions according to the optimal policy.

These are six plots which show how many times the driving agent took the optimal action according to the optimal policy within the last 40 iterations.