Project 4: Reinforcement Learning

Train a Smartcab to Drive

The setting

In the project “Train a Smartcab to Drive”, a smartcab operating in an idealized grid-like city is given. There are traffic lights at each intersection and other cars present. The smartcab gets a reward for obeying traffic rules and a penalty for not obeying traffic rules or causing an accident. Goal of this project is to implement a learning agent for the smartcab that should learn an optimal policy for driving on city roads, obeying traffic rules correctly, and trying to reach the destination within a goal time based on the rewards and penalties it gets.

Step 1 - Implement a basic driving agent

In the first step, I let the smart cab take a random action out of the possibilities of:

- doing nothing (state ‘None’)
- driving forward (state ‘forward’)
- turning left (state ‘left’)
- turning right (state ‘right’)

In this step, the smartcab does not learn from the results of it’s actions and has unlimited time to reach the goal.

I put the printed output of my test run into `'output_first_text.txt'`.

In [279]:
```
def read_in_my_text(file_name, print_lines):
    with open (file_name, "r") as myfile:
        output = myfile.read()
    count_reached = output.count('Primary agent has reached destina
tion!')
    count_aborted = output.count('Trial aborted.')
    lines = output.split('\n')
    print "Rate of reaching the destination {}".format(count_reache
d*1.0/(count_reached + count_aborted))
    if print_lines == True:
        print "Last 3 lines:"
        for line in lines[-3:]:
            print line
```

In [280]:
```
read_in_my_text("outputs/out_1_1.txt", False)
```

```
Rate of reaching the destination 0.19801980198
```

In [281]:
```
read_in_my_text("outputs/out_1_2.txt", False)
```

```
Rate of reaching the destination 0.25
```

In [282]:
```
read_in_my_text("outputs/out_1_3.txt", False)
```

```
Rate of reaching the destination 0.2
```

In [283]:
```
read_in_my_text("outputs/out_1_4.txt", False)
```

```
Rate of reaching the destination 0.23
```

In [284]:
```
read_in_my_text("outputs/out_1_5.txt", False)
```

```
Rate of reaching the destination 0.16
```

The smartcab reaches the destination in time in around 20% of the trials. If the smartcab reaches the destination is a question of chance. The actions are chosen randomly, independent of the destination and the received rewards.

Step 2 - Identify and update state

For the states, I chose a tupel of `'next_waypoint'`, `'light'`, `'oncoming'` and `'left'`. I didn't take `'right'` into account, because no matter what our smartcab is going to do, the traffic from the right side has no influence on it.

Step 3 - Implement Q-Learning

In this step I implemented the q-learning algorithm with a learning rate of 0.5 and a discount factor of 0.5. The next action is always chosen based on the best estemate based on the current state. I didn't use an exploration rate.

In [285]:
```
read_in_my_text("outputs/out_3_1.txt", True)
```

```
Rate of reaching the destination 0.911764705882
Last 3 lines:
Environment.act(): Primary agent has reached destination!
Environment.act(): Primary agent has reached destination!
Environment.act(): Primary agent has reached destination!
```

In [286]:
```
read_in_my_text("outputs/out_3_2.txt", True)
```

```
Rate of reaching the destination 0.93137254902
Last 3 lines:
Environment.act(): Primary agent has reached destination!
Environment.act(): Primary agent has reached destination!
Environment.act(): Primary agent has reached destination!
```

In [287]:
```
read_in_my_text("outputs/out_3_3.txt", True)
```

```
Rate of reaching the destination 0.06
Last 3 lines:
Environment.step(): Primary agent ran out of time! Trial aborted.
Environment.step(): Primary agent ran out of time! Trial aborted.
Environment.step(): Primary agent ran out of time! Trial aborted.
```

In [288]:
```
read_in_my_text("outputs/out_3_4.txt", True)
```

```
Rate of reaching the destination 0.970297029703
Last 3 lines:
Environment.act(): Primary agent has reached destination!
Environment.act(): Primary agent has reached destination!
Environment.act(): Primary agent has reached destination!
```

In [289]:
```
read_in_my_text("outputs/out_3_5.txt", True)
```

```
Rate of reaching the destination 0.950980392157
Last 3 lines:
Environment.act(): Primary agent has reached destination!
Environment.act(): Primary agent has reached destination!
Environment.act(): Primary agent has reached destination!
```

Interestingly, the smartcab reaches the destination quite often in time. There seem to be some cases, when the smartcab get's stuck and doesn't reach the destination in time.

Step 4 - Enhance the driving agent

I played around with the learning rate and calculated it using the time step. I stayed with a discount factor of 0.1. I tried several exploration rates, but that lead to not reaching the goal more often.

In [290]:
```
read_in_my_text("outputs/out_5_1.txt", True)
```

```
Rate of reaching the destination 0.950495049505
Last 3 lines:
Environment.act(): Primary agent has reached destination!
Environment.act(): Primary agent has reached destination!
Environment.act(): Primary agent has reached destination!
```

In [291]:
```
read_in_my_text("outputs/out_5_2.txt", True)
```

```
Rate of reaching the destination 0.98
Last 3 lines:
Environment.act(): Primary agent has reached destination!
Environment.act(): Primary agent has reached destination!
Environment.act(): Primary agent has reached destination!
```

In [292]:
```
read_in_my_text("outputs/out_5_3.txt", True)
```

```
Rate of reaching the destination 0.94
Last 3 lines:
Environment.act(): Primary agent has reached destination!
Environment.act(): Primary agent has reached destination!
Environment.step(): Primary agent ran out of time! Trial aborted.
```

In [293]:
```
read_in_my_text("outputs/out_5_4.txt", True)
```

```
Rate of reaching the destination 0.98
Last 3 lines:
Environment.act(): Primary agent has reached destination!
Environment.act(): Primary agent has reached destination!
Environment.act(): Primary agent has reached destination!
```

In [294]:
```
read_in_my_text("outputs/out_5_5.txt", True)
```

```
Rate of reaching the destination 0.92
Last 3 lines:
Environment.act(): Primary agent has reached destination!
Environment.act(): Primary agent has reached destination!
Environment.act(): Primary agent has reached destination!
```

In addition to the changes I made to the learning rate, I changed the initialization from my q table from initializing it with zeros to using random values between 0 and 4. With my first attempt the smartcab got stuck most of the time.

My final agent finds a close to optimal policy. The rewards increase with time and the smartcab reaches the destination in time most of the time.