Training a smartcab with q-learning

April 17, 2016

0.1 Implement a basic driving agent

In your report, mention what you see in the agent's behaviour. Does it eventually make it to the target location?

When I set the agent to choose a random action, it travels around randomly, disobeying the rules of the road and receiving many penalties. It eventually does make it to the destination. I would call this behaviour highly explorative.

0.2 Identify and update state

Justify why you picked these set of states, and how they model the agent and its environment.

I initially selected a large state space that used all available information. This included

- Light colour
- Car direction at each intersection
- Next waypoint
- Deadline

I quickly realised that this state space has more information than is required for the agent to act in an acceptable fashion, so I refined it. Firstly I dropped the deadline as it has the largest impact on the state space size, and then I refined the information that required from the light and the intersection traffic . The state space became

- W, the direction of the next waypoint. Values (wL|wR|wF|wX)
- F, whether it's legal for the car to go forward. Values (F|xF)
- L, whether it's legal for the car to turn left. Values (L|xL)
- R, whether it's legal for the car to turn right. Values (R|xR)

The above state space is small enough to allow for quick learning, and although I have implemented logic for legal turns, the agent will still need to learn how to use this information.

0.3 Implement Q-Learning

What changes do you notice in the agent's behaviour?

I implemented q-learning and made the agent chose the action that correlated with the highest q-value in the agents current state (s).

In doing this the agent would often choose the same action, which generally was to turn right or stay still. This is because the agent would collect a positive reward from these actions, which would reinforce this same choice by increasing the q-values for them. When this happens we can say that the agent has found a local maximum.

Here is the output after running the agent for 100 trials with the following variables

Gamma = 0.5; Learning Rate = 0.5; Chance of random action = 0.0; Q-value initialisation value = 0.0

```
Complete:g:+0.500, lr:+0.500, rv:+0.000, qi:+0.000, dest:0, p:4, t:47.116, #finished:1
--------------------------------------------------------------------------------------------------------------
#Trials:100, Trial no:100, Step:42, Completions:0
Trial rewards:[0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5,
0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5,
0.5, 0.5, 0.5, 0.5]
                  State|,                                          Q-values |, Policy
   ('wF', 'L', 'R', 'F'), ['N:+0.000', 'f:+0.000', 'l:+0.000', 'r:+1.120'], ['right']
 ('wF', 'xL', 'R', 'xF'), ['N:+0.000', 'f:+0.000', 'l:+0.000', 'r:+1.126'], ['right']
('wF', 'xL', 'xR', 'xF'), ['N:+0.000', 'f:+0.000', 'l:-0.220', 'r:+0.000'], [None, 'forward', 'right']
   ('wL', 'L', 'R', 'F'), ['N:+0.000', 'f:+1.119', 'l:+0.000', 'r:+0.000'], ['forward']
 ('wL', 'xL', 'R', 'xF'), ['N:+0.000', 'f:-0.191', 'l:+0.000', 'r:+1.069'], ['right']
('wL', 'xL', 'xR', 'xF'), ['N:+1.238', 'f:+0.000', 'l:+0.000', 'r:+0.000'], [None]
   ('wR', 'L', 'R', 'F'), ['N:+2.000', 'f:+0.000', 'l:+0.000', 'r:+0.000'], [None]
  ('wR', 'xL', 'R', 'F'), ['N:+0.000', 'f:+0.000', 'l:+1.325', 'r:+0.000'], ['left']
 ('wR', 'xL', 'R', 'xF'), ['N:+2.000', 'f:-0.158', 'l:+0.000', 'r:+0.000'], [None]
('wR', 'xL', 'xR', 'xF'), ['N:+2.000', 'f:-0.002', 'l:+0.000', 'r:+0.000'], [None]
```

We can see from this output that the trial rewards are all 0.5, and that the agent did not get to it's destination because there would be a higher reward at the end of this list.

We can see that the agent did search the state space a little, this is indicated by the number of states in the q-matrix list. Here we can see that 10 states out of a possible 24 have been visited.

On the far right we can see the policy of the agent, which describes the action the agent will chose if it is in that state. From the output it shows that the agent would

- Turn right in 3 out of 10 states
- Do nothing in 4 out of 10 states
- Incur a penalty in state ('wR', 'xL', 'R', 'F') as it should not turn left
- Probably incur a penalty in the state ('wF', 'xL', 'xR', 'xF')
- Choose the wrong direction (i.e. not towards the next waypoint) in 5 out of 10 states.

To summarise, the agent did not learn to reach the destination, but it did learn to avoid getting penalties.

0.4 Enhance the 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?

I think the optimum policy is for an agent is to reach the destination with a minimal number of penalties.

To have a better chance of achieving this, I used a number of independent variables to alter the way the agent behaved.

Gamma variable (g): Used in the Bellman equation to control the amount of information that is used from finding the max s_prime q-value. Range $0.0 <= g < 1.0$

Learning rate variable (lr): Used in the Bellman equation. This affects how much new information is stored over old information in the q-matrix. Range $0.0 <= lr < 1.0$

Q-initialisation variable (qi): The value is used to initialise new state space in the q-matrix. This affects how likely an agent is to explore the state space, as higher initial q-values would make the agent choose the action to try these new states. The range for this could be anything, but with our problem a range of $-5.0 <= qi < 5$ will suffice.

Random action variable (rv): A value that allows for random actions to happen. E.g. a random action value of 0.2 means that 20% of the time the agent will chose an action randomly. This affects how explorative the agent is and helps an agent to step out of local maximums. Range $0.0 <= rv < 1$

0.4.1 Choosing variable values

There are many ways to discover the best values for these variables, and I have opted to run the agent for a number of samples with random values for the variables. The values chosen will be randomly distributed within the ranges discussed above.

After taking the samples I will select the top 5% based on a score function, and use the means of this dataset for the variables in my optimal agent.

For the scoring function, I have chosen this formula,

$$s = \sum(c) - \left(\frac{\sum(p)}{t}\right)^2$$

where the score (s) is the sum of the number of times the agent reached the destination (c), minus the average number of penalties squared. Where p indicates the number of penalties and t is the number of trials.

Each sample and score will be calculated over 100 trials and I collected 6000 samples, here are the results.

```
In [46]: import pandas as pd
         import numpy as np
         import re
         pd.set_option('precision', 5)
         np.set_printoptions(precision=2, suppress=True)
         import os

         directory = "."
         filenames = [os.path.join(directory, f) for f in os.listdir(directory) if re.match(".+t100_s10

         df = pd.DataFrame()
         for filename in filenames:
             # print "Loading file:%s" % filename
             df_csv = pd.read_csv(filename)
             df = pd.concat([df, df_csv], ignore_index=True)

         # Calculate the score as described above
         df["score"] = df["r"] - (df["p"] / 100) ** 2

         # Take the top 5% of scores
         df_best = df[df.score > df.score.quantile(.95)]

         # Descrive the data
         print df_best[['g', 'lr', 'rv', 'qi']].describe()
```

```
g          lr          rv          qi
count  300.00000   300.00000   300.00000   300.00000
mean     0.31003     0.52519     0.10123     2.47571
std      0.20931     0.28875     0.06415     1.98333
min      0.00042     0.00190     0.00177    -4.81933
25%      0.14180     0.29725     0.05104     1.72514
50%      0.28801     0.52547     0.09418     2.93091
75%      0.45664     0.77819     0.14063     3.87140
max      0.88973     0.99812     0.32781     4.99096
```

0.5 The optimal agent

I ran the agent with the variables set to the means taken from the above statistics.

- Gamma = .31003

- Learning rate = 0.52519
- Random chance = 0.10123
- Q-init value = 2.47571

The output of this agent is shown here.

```
Complete:g:+0.310, lr:+0.525, rv:+0.101, qi:+2.476, dest:97, p:40, t:23.573, #finished:1
-----------------------------------------------------------------------------------------------------------
#Trials:100, Trial no:100, Step:14, Completions:97
Trial rewards:[1, 1, 1, 1, 1, 2, 2, 2, 1, 1, 1, 1, 12]
                  State|,                                          Q-values |, Policy
   ('wF', 'L', 'R', 'F'), ['N:+2.753', 'f:+8.463', 'l:+2.713', 'r:+2.145'], ['forward']
 ('wF', 'xL', 'R', 'xF'), ['N:+2.137', 'f:+0.478', 'l:+0.845', 'r:+1.854'], [None]
('wF', 'xL', 'xR', 'xF'), ['N:+2.246', 'f:+2.184', 'l:+0.692', 'r:+2.310'], ['right']
   ('wL', 'L', 'R', 'F'), ['N:+2.161', 'f:+1.739', 'l:+4.429', 'r:+1.578'], ['left']
 ('wL', 'xL', 'R', 'xF'), ['N:+3.720', 'f:+0.394', 'l:+0.473', 'r:+1.916'], [None]
   ('wR', 'L', 'R', 'F'), ['N:+2.515', 'f:+1.780', 'l:+1.841', 'r:+6.701'], ['right']
 ('wR', 'xL', 'R', 'xF'), ['N:+2.476', 'f:+0.636', 'l:+2.048', 'r:+9.456'], ['right']
('wR', 'xL', 'xR', 'xF'), ['N:+2.476', 'f:+2.476', 'l:+1.163', 'r:+2.476'], [None, 'forward', 'right']
```

From these results we can see

- From the top row, it shows the agent reached the destination in 97 out of 100 trials, (dest:97).
- The agent was penalised 40 times, (p:40) in 100 trials.
- In the last trial, the agent did not receive any penalties as the Trial_rewards are all positive.
- In the last trial, the agent did reach the destination as we can see a reward of 12 as the last item.
- The agent made it to 8 different states out of the possible 24 as there are 8 rows in the q-matrix.

From these statistics we can say the agent has performed extremely well and has learnt very quickly, but has it learnt the optimum policy?

```
State          Policy                  Notes
wF  L  R  F => forward                 OK
wF xL  R xF => None                    OK
wF xL xR xF => right                   Penalties!
wL  L  R  F => left                    OK
wL xL  R xF => None                    OK
wR  L  R  F => right                   OK
wR xL  R xF => right                   OK
wR xL xR xF => None, forward, right    Possible penalties!
```

We can see that the agent does not conform to the optimum policy as it is able to collect penalties.

In [ ]: