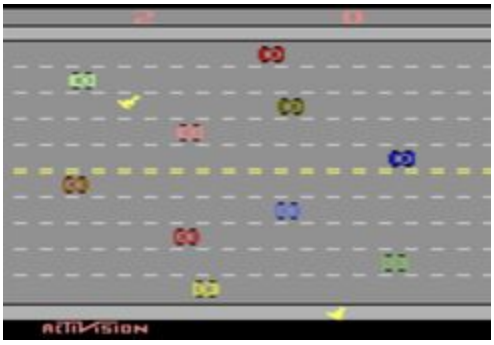


Why Did the Chicken Cross the Road

Q- learning - Freeway

Marissa Montano and Jorge Moreno
CSCI 3202 - Intro to AI
CU Boulder - Spring 2019
Practicum

Problem

Can we get an agent using, reinforcement learning, to beat the average and high score of its creators? The game we are trying play is the Atari game Freeway. The game is simple: in 2 minutes and 16 seconds get the most chickens across the street as possible. The rules are that you can only move forward, back, or stay in place and if you get hit by a car you don't die, you just delay progress. These actions were represented as: 0-no-op, 1-forward, 2-backward. The problem we are trying to attack is one of policy learning. Our goal is to train an agent to learn the optimum policy of the game state in order to get an average score at or above 17 and a high score of at or greater than 19. If we can beat these baseline goals we want to try to beat the highest human score of 35.

Methods

The method we used to solve this markov decision problem was Q-learning. This is a good method to use because we had an agent, a state space, some set of actions, and a goal that gave a reward when completed. In order to get the max reward, we had to plan out an optimum policy. To do this we needed some way of influencing our current action based on future reward. Q- learning with the greedy epsilon approach worked specifically well to solve this problem.

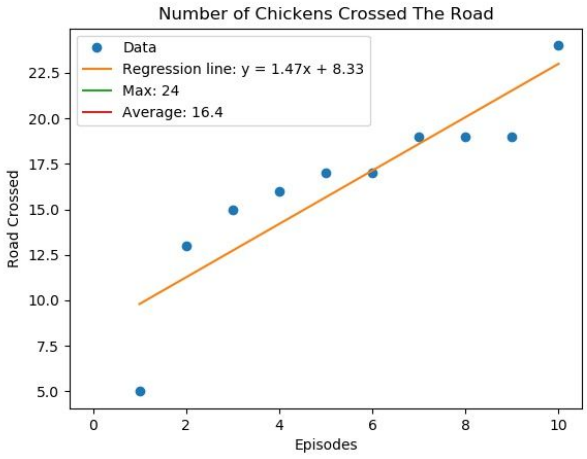
For Q-learning the idea is to basically start with empty Q-states and slowly update them as we begin to explore, that way the agent knows the chance it has of either getting a reward or getting a punishment. The main training loop we used was to pick an action and update the Q-space based on the rewards, learning rate, and old Q-values. After you do this for a couple episodes (we did 10, 25, 100, and 1000 episodes) the agent performed better than it did in the previous episodes because it has started to learn the best path to take each time.

The important approaches to solving our problem was understanding and manipulating the parameters. To update the Q-states we had to use the Bellman equation that needed a learning rate alpha and a discount rate gamma (Q(s, a) = alpha * (r(s, a) + gamma * max Q(s', a') - Q(s, a))) and we had an exploration rate epsilon that directly affect the action a. The alpha parameter helped up decide how much we were going to allow ourselves to learn (ie. exploit old data or introduce new possibilities). The gamma parameter is the discount factor and is used to remind our agent of our ultimate goal (ie. take the closest reward now or wait for a bigger reward later). The epsilon parameter is used to help decide what actions to make (ie. be riskie and go explore or play it safe and only go where we know its safe). The trick to making our application work was annealing the parameters. We started with a high alpha and epsilon in order to quickly learn where and how they could move and over time we decreased them to exploit the already learned optimal path. For our gamma parameter we really did not have to mess with it too much because

giving rewards was nearly impossible. Our state space was extremely hard to dissect and read because our state space was a matrix with 128 observations ranging from 0 to 255 and there was no source code, blog posts, or tutorials saying what each state meant. So we couldn't do much in terms of reward or punishment based off of the output of our state space. A couple of ideas we wanted to implement was a punishment for getting it, a punishment for taking too long to make a point and giving a reward when the chicken crossed the road.

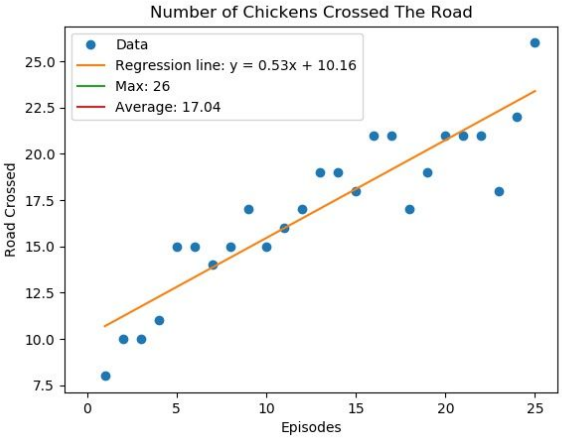
Results:

After implementing our Q learning algorithm, we were able to successfully reach our initial goals of having an average of 17 chickens cross the road and a max that was greater than 19. The way we implemented this was by adding trials and a reward system. As explained above, each trial was 2 minutes and 16 seconds long. At first, we were unable to reach the end goal since there were no rewards other than reaching the end. This was impossible to reach because of the size of our environment. It could have learned without help if we had a lot of runs (5000+) as well as time (1000 runs took around 30 minutes to train). In order to remove this roadblock, we gave the agent a reward of +1 for going forward, on top of the state reward, which it got +1 every time it reached the end. We also tried some other reward systems but they will be discussed with each image below. We also gave our graphs regression lines as well as max and average for a better visual representation.



(Figure 1,Left, 10 Trials with Forward Reward +1)

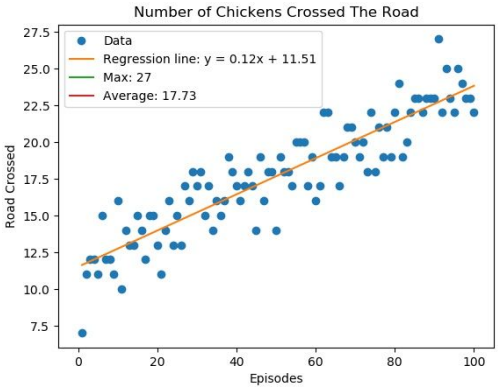
For our first trial run, we ran our Q learning agent a total of 10 episodes. At first, our agent had a rough time crossing the road. We can see that it crossed the road 5 times in the first 2 minutes and 16 second. After that though, it made huge improvements getting a max of 24, already beat our record there, and an average of 16.4. We can do better



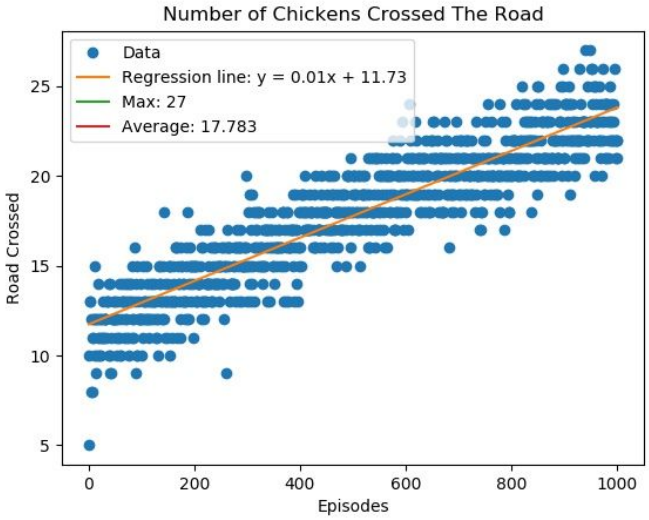
(Figure 2, Right, 25 Trials with Forward Reward +1)

For our next run, we decided to step it up and give the agent more time to learn. We saw from the previous test that our data was not flat lining. Therefore upping the trials seemed like a good solution. From this run, we can see that the max improved by 2 more, as well as our average. We

were able to surpass the our average goal by .04. With that said, we thought we could do better.

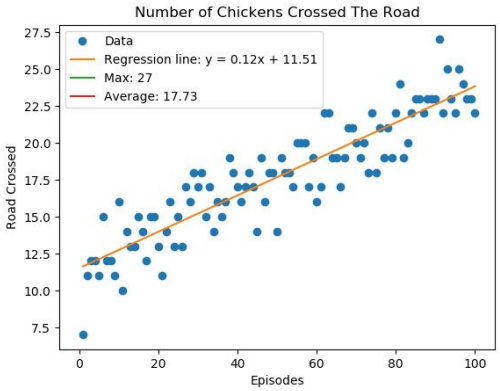


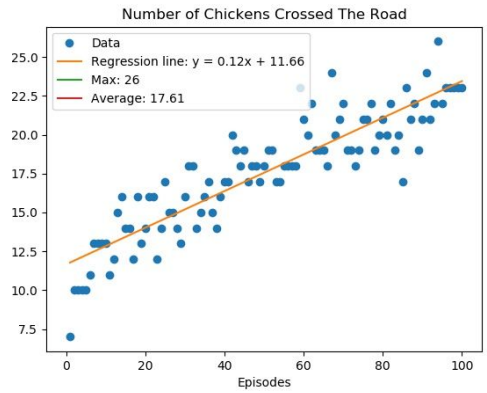
(Figure 3, Left, 100 Trials Forward Reward +1)
As we rose our trials, our Q learning took longer to run. This 100 trial run took about 5-10 minutes to finish. The same reward system was kept and we saw that our data was not incrementing as fast as the other two. With that said, we believed that we could surpass 27 roads crossed as our max so we did one more big run for the fun of it.

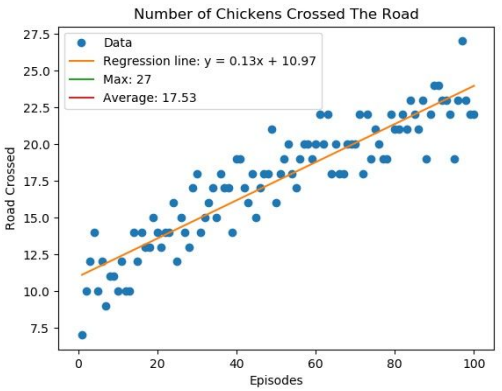


(Figure 4, Right, 1000 Trials, Reward +1 Forward)
Our last Q learning algorithm we ran consisted of 1000 trials. Each trial again was 2 minutes and 16 seconds long. From the data, we can see that our algorithm did indeed start flat lining at a max of 27 roads crossed and an average between 17.7 and 17.8 roads crossed. This process took 30-45 minutes long. Unfortunately, we were unable to reach the max score we read online (35)

<u>The following three figures where the three different reward systems tried to increment the number of roads crossed with a trial of 100 runs</u> (Left to Right: Figure 5, Figure 6, Figure 7)







We decided to run the 100 trials for each of the three different kind of reward systems because they did not take too long to run and the results were pretty decent. Figure 5, was run

with a reward of +1 for going up. Figure 6, was the run with rewards of +1 for going up and -2 for going down. Finally, Figure 7 was run with +2 going forward, -2 going down, and + 1 for staying.

For Figure 6, we tried to teach our agent that going down was a bad choice. We tried to make it learn that it would be safer to do a no-op, stay still, or go up rather than hit a car and be returned down.

For Figure 7, we tried to enforce from the previous reward system. We upped the reward for going up to +2, going down -2 and doing a no-op as +1. Our intention was to make the agent learn that no matter the choice it chose, going down was the worst possible action to take.

In the end, these two reward systems did not seem to improve our run max and average roads crossed. We decided to stick with the original +1 for going forward.

Comparing all of this to someone else was kind of hard. This is because it does not seem that the Gym AI platform is widely known and the atari game we did was freeway-ram-v0. With that said, we were able to find two markers to hit from previous users. 35 was the max user score someone got from using keyboard while 17 was what we got an average when we were first test running everything.

Conclusion:

In the end, we were able to beat our baseline goals. The max score we wanted to beat was 19 and we were successfully able to get 27 even though we did see a 28 on one of our first attempts. As for the average, we did surpass it barely by ~0.7 points. Even though we didnt beat our ultimate goal, we were able to do what we set out to do and ended up reaching our goal using the method we initially thought would work: Q-Learning.

For future improvements, we were thinking about changing our variables. I.E. changing our alpha learning rate, our discount rate, epsilon starting point, and minimum so that it would always have a constant search range. Also, we were unable to understand how to read the observation space given to us from the environment therefore we couldn't use that data to make any meaningful changes in the way we gave or restricted the rewards. We tried to check if there was a given state dedicated to getting hit by a certain vehicle but there was insufficient documentation on this version and not enough time to crack the code. The last big improvement we would like to try is adding in a convolutional neural network. After reading some literature it looks like if we really want big improvements in beating our goals for the game we would need to implement a DQN like DeepMind.

Resources:

1.) https://towardsdatascience.com/reinforcement-learning-with-openai-d445c2c687d2

2.) https://moodle.cs.colorado.edu/pluginfile.php/157112/mod_resource/content/1/IntroToAI_10_proper.pdf

3.) https://gym.openai.com/envs/#atari

4.) https://towardsdatascience.com/simple-reinforcement-learning-q-learning-fcddc4b6fe56

5.) https://medium.com/machine-learning-bites/machine-learning-reinforcement-learning-markov-decision-processes-431762c7515b

Extra:

```
[ 0 91 167  0 13 15  0 255 74 30 12  6  0  8 86  6 255 255
  7  7 26  0 255 80 192 48 32 144 16 32 176 64 208  1  2  2
  0  0  0  1  1  1  0 52 120 243 69 42 16 245 55 178 246 80
 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80
 80 80 80 80 80 26 216 68 136 36 130 74 18 220 66 189 247 80
247 122 247 80 247 80 247  8 247  0 247  0  0  1  0  4 70  0
 63 119 52 77 154  5 82 108 41 97 16 16  5 63 80 255 87 246
 75 244]

[ 0 93 167  0 14 15  0 255 74 30 12  6  0  8 86  6 255 255
  7  7 26  0 255 80 192 48 32 144 16 32 176 64 208  4  0  0
  0  0  0  1  2  3  3 36 120 243 53 10 48  5 71 194  6 80
 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80
 80 80 80 80 80 26 216 68 136 36 130 74 18 220 66 189 247 80
247 122 247 80 247 80 247  8 247  0 247  0  0  1  0  4 69  0
 64 119 52 78 156  3 81 107 40 96 48 16  5 64 80 255 87 246
 75 244]
```

The above image is an example of our state space. This is two back to back state spaces after an action is made from the chicken. The only two significant numbers we were able to decode where the one in red that gave us the height position of our chicken after an action (no-op, up or down) and the yellow which gave us the total amount of times the chicken had crossed the road.