# Homework 6:  Reinforcement Learning

CS 545:  Machine Learning, Winter 2017

Ben Wilson

## Introduction:

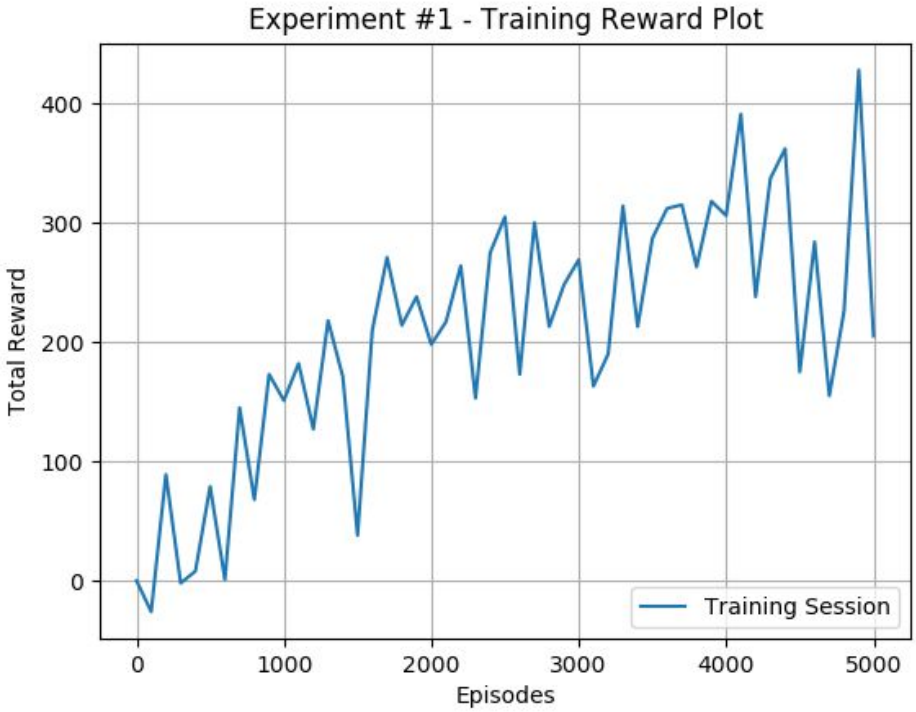
Reinforcement learning  uses a system of rewards and punishments to teach an agent an appropriate behavior. In this assignment we were asked to simulate the training of Robby, a can collecting robot. Robby can see it's current location and the locations to it's immediate left, right, front, and rear. It also has the options of attempting to pick up a can at it's current location or moving to an adjacent location. Robby is rewarded when it successfully picks up a can and is punished whenever it tries to pick up a can on an empty location or whenever it runs into a wall.

In order to be trained properly Robby needs to explore the environment and gain some intuition on how to behave in different situations. This is implemented with a Q-Matrix that keeps track of which action in a given state leads to the highest reward. Robby also uses epsilon-greedy action selection. This means that there is a probability, epsilon, that Robby will "explore" by making a random move. In this assignment, epsilon starts at 100% and incrementally moves towards 10%, i.e. he takes a lot of risk at first, but over time is risk averse.

# Experiment 1:

For the first experiment we were asked to simulate training Robby over 5000 episodes, each episode having 200 steps. The learning rate is set to 0.2, the discount factor is 0.9, and epsilon starts at 100% and is decreased by 1% every 50 episodes until it reaches a minimum of 10%.

Robby's environment is a 10x10 grid with cans placed on half of the locations. Robby's total reward is based on the following: 10 points for collecting a can, -1 point for attempting to collect a can on an empty location, and -5 points for crashing into a wall. A perfect score of 500 would be achieved by collecting 50 cans and never making a false attempt to collect a can or running into a wall. The environment is repopulated with cans so that they fill 50% of the spaces. The following plot shows the total reward received per episode for every 100th training episode.



Even though there's a noticeable amount of oscillation in the training reward plot, you can see that Robby progressively learns to maximize its reward.

After Robby has been trained, it's tested with the same settings (the Q-matrix in not updated) and the test average and test standard deviation are calculated to be the following:

**Test Average: 280.50**

**Test Standard Deviation: 84.42**

This would indicates that,on average, Robby is able to collect more than half of the cans each episode.

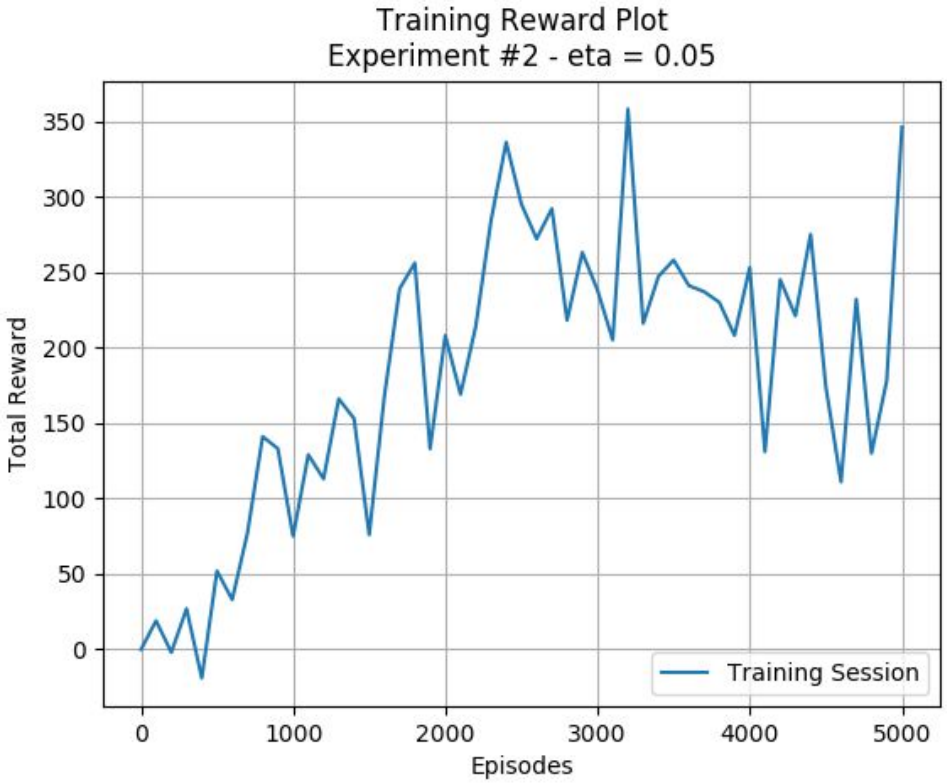
# Experiment 2: Experiment with the Learning Rate

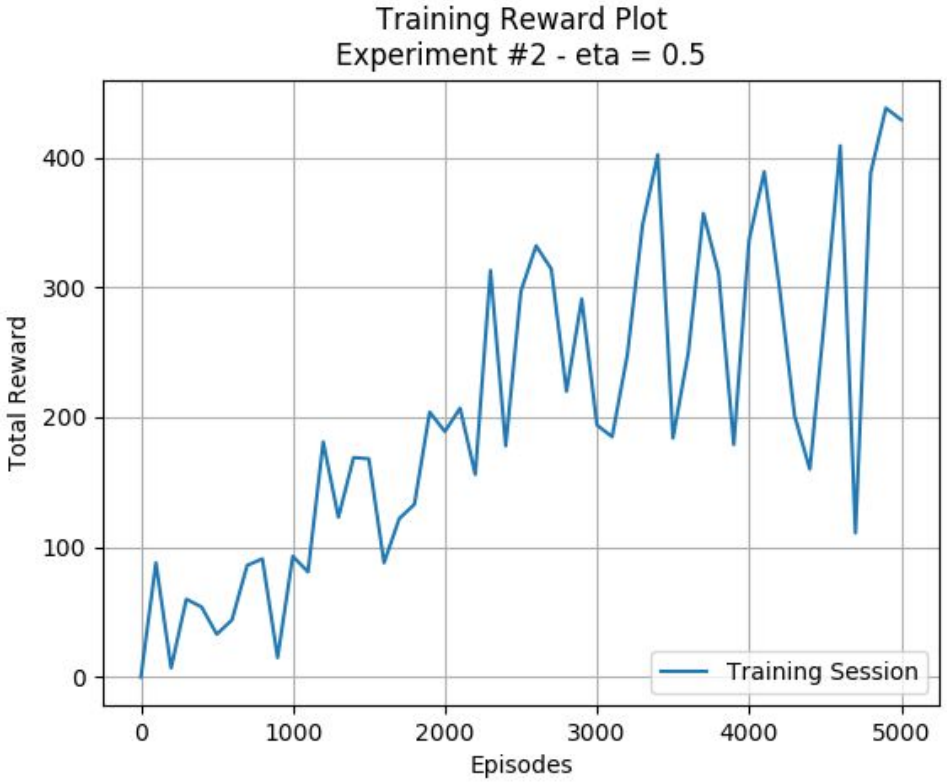
In the second part of this assignment, we were asked to see how altering the learning rate affects the results. Since Experiment #1 used a learning rate of 0.2, learning rates of 0.05, 0.5, 0.75, and 1.00 were chosen to get a wide range of possible results. The following table summarizes the results. It appears that making the learning rate too small or too large yields the worst results, but it isn't clear what intermediate value would work best. In this trial, we can see that using a learning rate of 0.50 had the largest test average. Further tests would be warranted to see if there is an ideal learning rate given the other parameters remain the same.

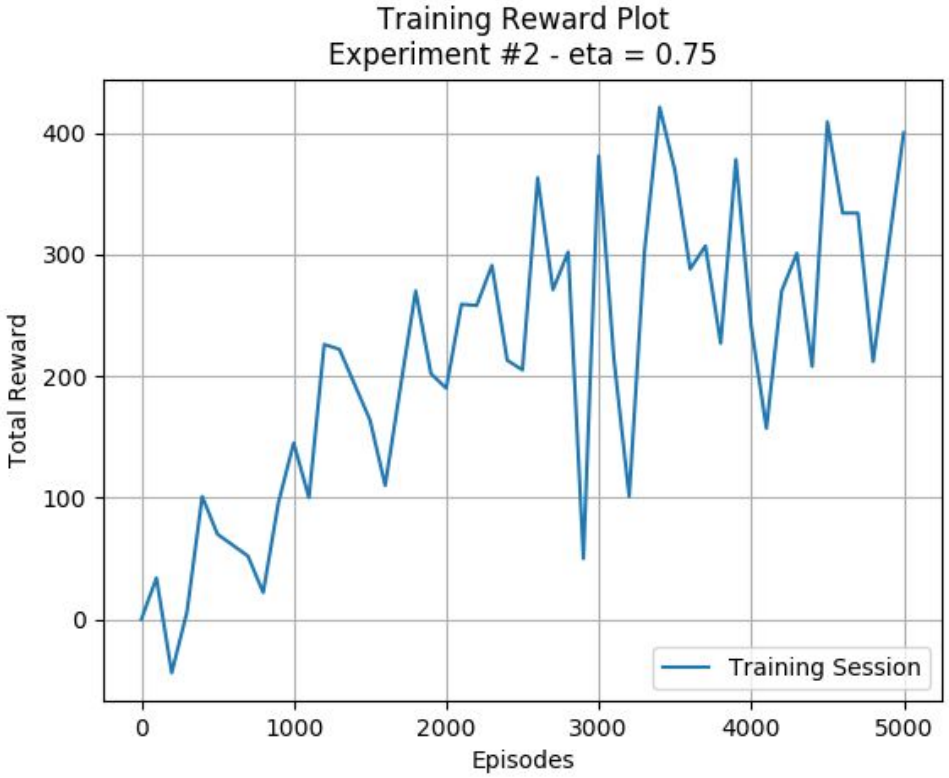
| eta | 0.05 | 0.50 | 0.75 | 1.00 |
|---|---|---|---|---|
| Mean | 279.69 | 288.89 | 282.12 | 268.01 |
| Standard Deviation | 92.57 | 84.50 | 82.50 | 78.48 |

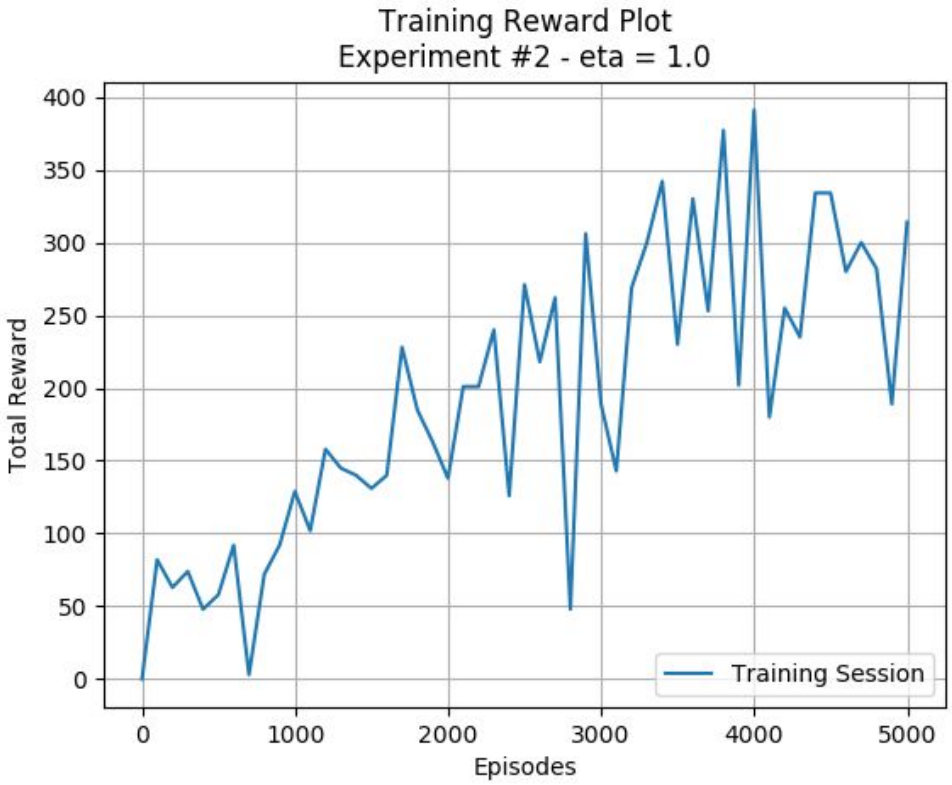
On the following pages are plots of the results from training Robby in Experiment #2. When eta is 0.05 we see that Robby learns at a somewhat linear rate for the first half of the training session, but then starts oscillating after 2500 episodes. We see a similar result when eta is 0.5, but with a shallower slope. Based on the training when eta=0.5, it's surprising that it had the best test mean.

Visually, it would appear that the best training session was when eta=0.75. We see a distinct linear trend up to roughly 3500 episodes (with a couple outliers) before some oscillations start. When eta=1.0 we see a linear slope with few oscillations, but increasing with a shallower slope than when eta=0.5 or eta=0.75. Based on the plot and test result, this would seem to indicate that setting the learning rate too high is worse that setting it too low.




Training Reward Plot
Experiment #2 - eta = 0.5
Total Reward
Episodes
Training Session
0
100
200
300
400
0
1000
2000
3000
4000
5000


Training Reward Plot
Experiment #2 - eta = 0.75
Total Reward
Episodes
Training Session
0
100
200
300
400
0
1000
2000
3000
4000
5000



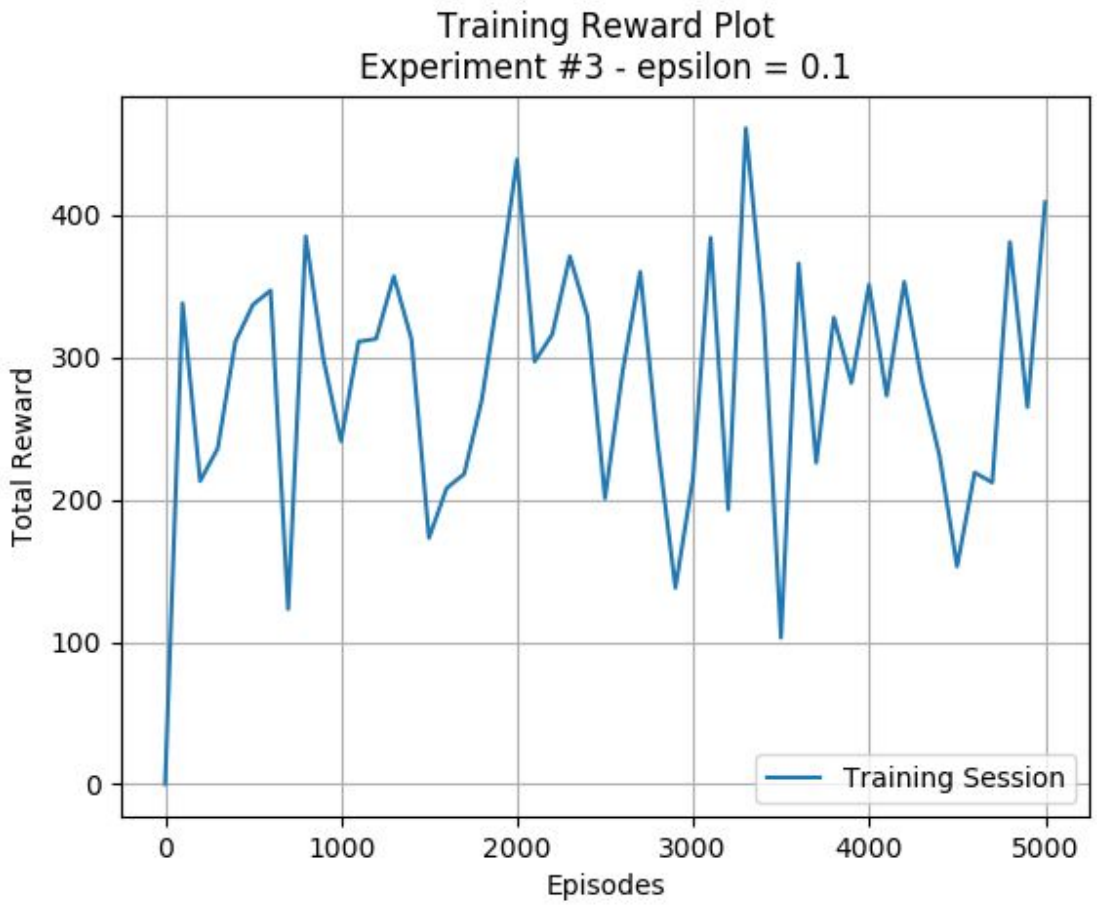

# Experiment 3: Experiment with Epsilon

This experiment has the same setup as Experiment #1 with the exception that the value of epsilon is kept constant throughout the training and testing.

We can see from the plot below that Robby's average total reward quickly jumps to roughly 300 and then oscillates for the rest of the training session. If epsilon is kept very low, Robby tends to be risk averse and may miss the best action had it been able to explore more and make more mistakes earlier. I believe this is why we see these oscillations in training.

**Test Average: 283.73**

**Test Standard Deviation: 84.65**



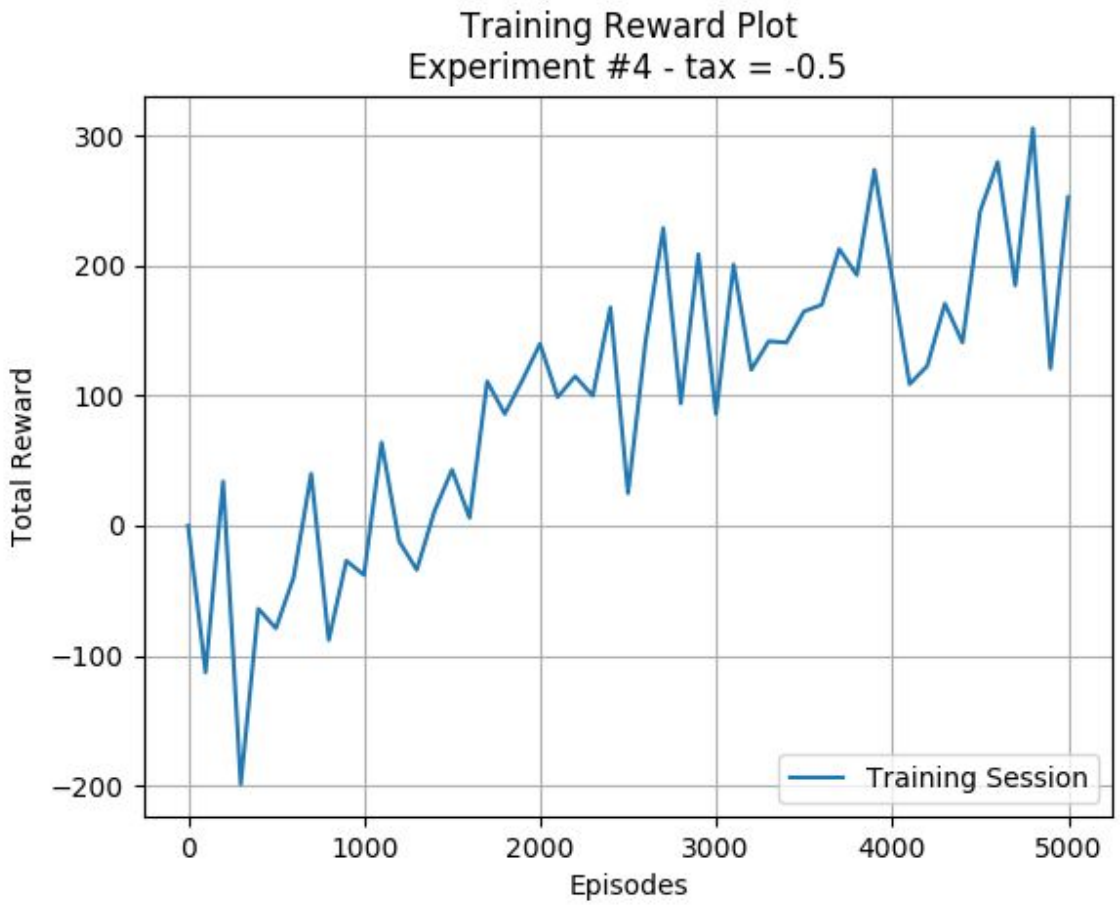

# Experiment 4: Experiment with Negative Reward for Each Action

This experiment is similar to Experiment #1 with the difference that an action tax of -0.5 is added to each move. Intuitively, this means that the average total reward would be lowered by 100 points each episode. And we see this reflected in the training session plot as well as in the testing average and testing standard deviation. In the training plot and in the test average we

see that the results are shifted down by roughly 100 points, while the testing standard deviation remains close to the standard deviation in Experiment #1 of 84.42.

**Test Average: 185.27**

**Test Standard Deviation: 85.46**




# Experiment 5: Choose Your Own Experiment!

For the last experiment the environment was changed from a 2D 10x10 square grid to a 3D 10x10x10 cube. The code needed to be updated to account for the addition of many more possible states and two more possible actions - move up and down. Initially this experiment was

just a scaled up version of Experiment #1, but an issue that came up was that if the number of steps was kept at 200 Robby would rarely hit a wall and would have plenty of low hanging fruit if the number of cans was kept at 50% fill. As a result, Robby learned very quickly to pick up cans and didn't have much of an issue with running into walls. This didn't seem that interesting.

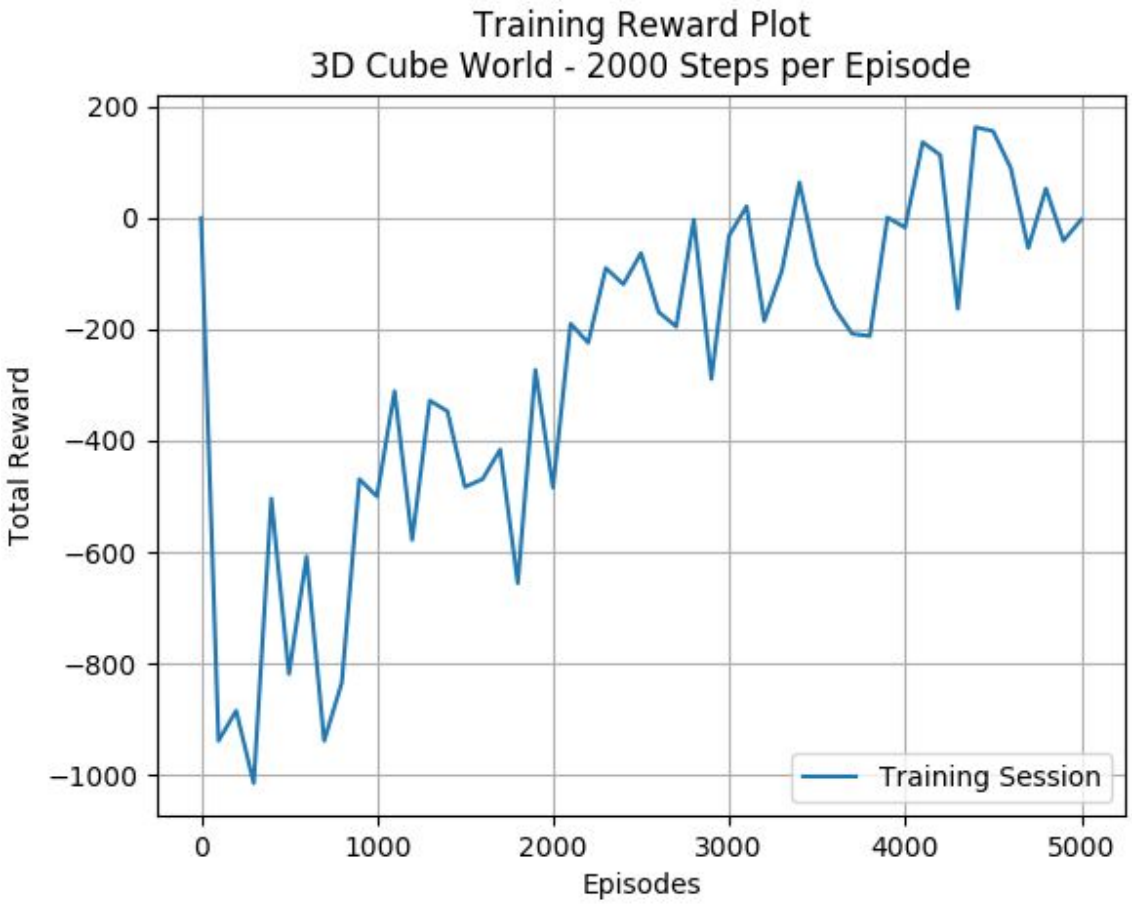
To make things more exciting, the number cans was reduced to 5% (fifty cans distributed over 1000 locations) and the number of steps was increased to 2000. This meant that Robby would have to collect the same number of cans as in Experiments 1-4, but in a larger and more complicated environment. The additional number of steps meant that Robby would have ample opportunity to explore the environment, make mistakes, and learn the best policy to maximize its reward.

The plot on the following page displays the result of the training session. We can see that Robby had a difficult time earning a large reward, but the total reward did consistently improve over the course of the training. At the end of the test session we see that Robby was, on average, able to learn a policy that resulted in a positive total reward.

**Test Average: 107.14**

**Test Standard Deviation: 84.86**

After thinking more about this experiment, it seems that increasing the dimension of the environment isn't that interesting of a problem. The results obtained in this experiment may have been more easily achieved by keeping the 2 dimensional case and adjusting other parameters to get the same effect.


Training Reward Plot
3D Cube World - 2000 Steps per Episode
Total Reward
Episodes
Training Session
200
0
−200
−400
−600
−800
−1000
0
1000
2000
3000
4000
5000