Michelle Ho (michho)   
Godwin Vincent (godwinv)  
CSE 415: Introduction to AI

Project Report

Feature-Based Reinforcement Learning for the Rubik Cube Puzzle

# Partners

Michelle worked primarily on the problem formulation and feature representation for learning, while Godwin worked with the Q-learning algorithm. Both partners verified each other’s work and collaborated on the feature-based machine learning with Q-learning implementation. Both Michelle and Godwin also worked together to optimize efficiency and speed of reaching the goal state of the problem.

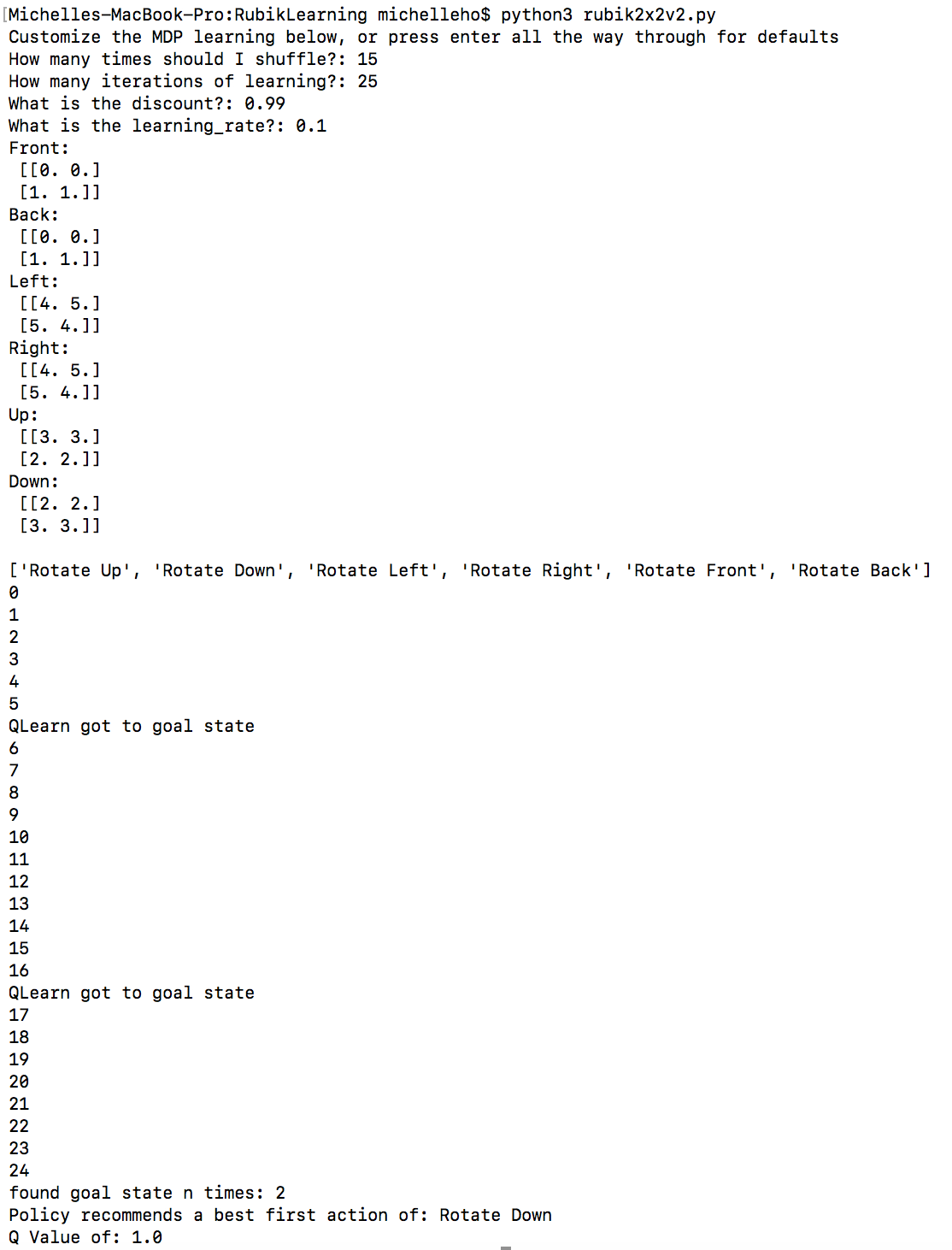
# Program Description

In this problem, we construct the problem formulation for the Rubik’s Cube puzzle and apply feature-based reinforcement learning in order to solve it. We formulated the problem with the following constraints: we will be solving a 2x2x2 Rubik’s cube and we will only be allowing 180 degree moves. Instead of exploring all the states like we would do in traditional Q-Learning, we used features to group states as we explore them.

# Applied AI Technique: SARSA

The technique we used is called SARSA. This is an abbreviation for “State–action–reward–state–action”. This is an algorithm for learning Markov decision process policy. We update the policy at each iteration using the information contained in the abbreviation: The current state, the action the agent chooses, the reward the agent gets for choosing that action, the state we get to by performing the action, and lastly the action the agent would choose from its new state. In addition to the SARSA technique we selected two features to use to represent our states. We initially decided the goal state should not require a color to be on a specific side and one could reach the goal state with all colors on a side, regardless of which side that was. That lead us to our first feature: the number of unique colors on each side. The higher the number the “worse” the state. We then realized that we also needed information on the intermediary states and decided to use adjacent pairs of the same color as our second feature. This counted the number of adjacent pairs of each color on a face. The higher the value, the “better” the state.

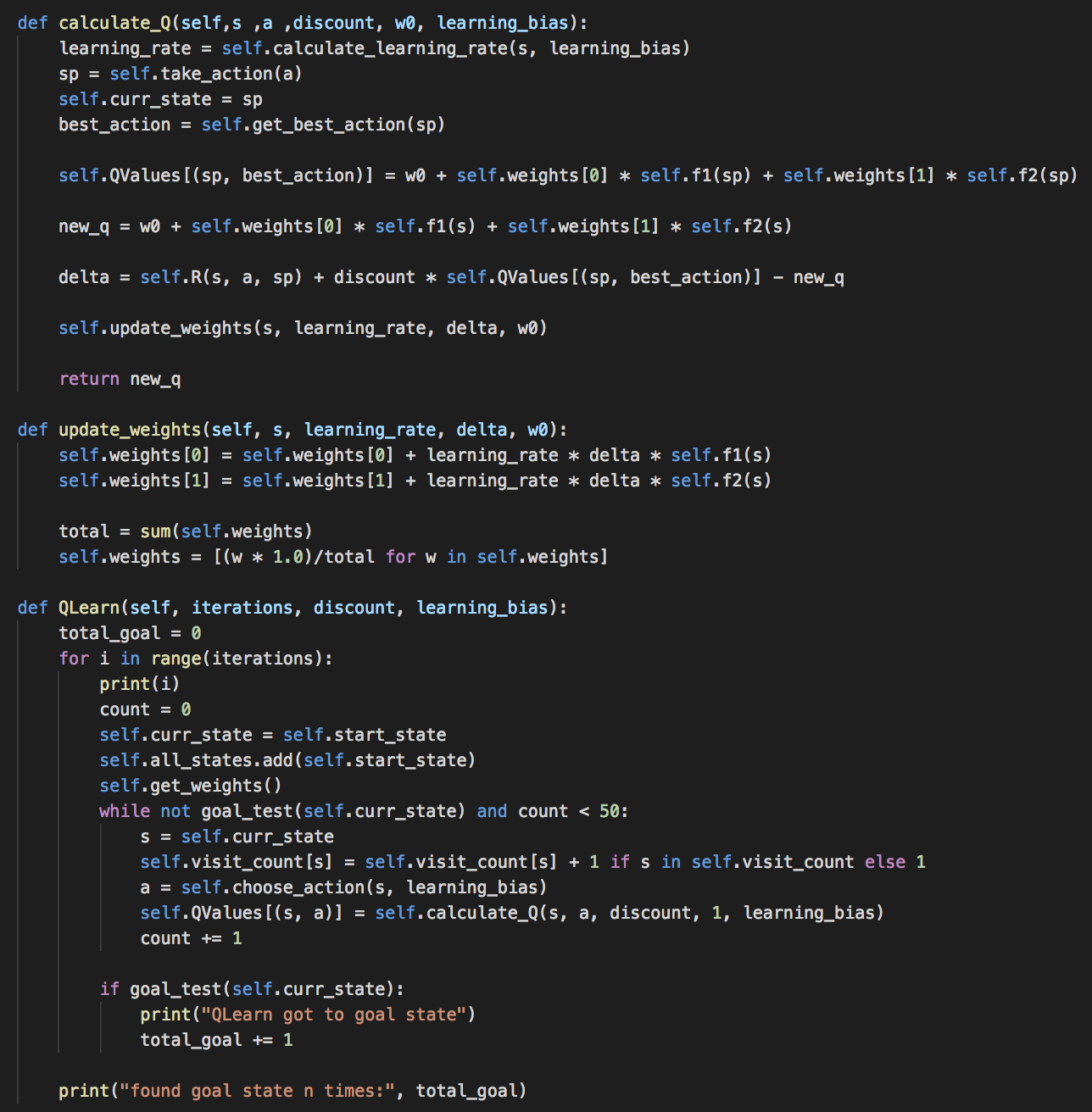
# Sample Session



# Demo Instructions

To run the code, run the file called ***rubik2x2v2.py***. From there, it will ask a series of questions to define the parameters of the problem solver. Specifically, the program will ask the user to define the number of times the cube should be shuffled, the number of iterations to be used for the Q-Learning algorithm, the discount to be used throughout the game, and the learning rate of the program.

# Code Excerpt



The code provided includes an implementation of feature-based Q-learning. First, the algorithm examines the current state and choose an action from that state. The state, **sp**, is the successor of the state **s** after applying the action **a**. The algorithm then uses a linear function of features to calculate the Q value in SARSA. Specifically, the Q value is computed by summing the initial weight and the weighted sum of the two features described above. At each iteration, the weights are updated by the Q-learning algorithm and the current state and action are updated with the future state, **sp**, and the action taken. This process is repeated for the number of iterations defined by the user.

# Lessons Learned

Throughout this project, we were able to reinforce essential topics covered this quarter and learn new techniques. Specifically, we were able to strengthen our understanding of problem formulation for a Rubik’s. This included making important decisions on how we wanted to represent our problem state, operators, and goal state, which paved the way for the entirety of our project.

In addition, we were also able to understand the differences between Q-learning and A-Star. Initially, when beginning this project, we attempted to solve the problem using A-Star and devise a policy from that specific algorithm. A-Star required generating the state space of the problem and always resulted in a solution to the problem. When moving forward to feature-based Q-learning, we learned that the representation of the state was extremely important. Specifically, devising features to represent our data was a challenge, as we wanted to be able to recognize unique, yet defining patterns in which state representations could be grouped together.

# Moving Forward

Given more time to add to our program, we would have liked to implement a solution to solving a 3x3x3 Rubik’s cube with 90 degree moves, as well as define more features to better represent groups of states. With an increased state space given by a 3x3x3 Rubik’s cube, feature representation would become more critical in grouping states together.

# Citations

1. Artificial Intelligence, Poole & Mackworth (LCI, UBC, Vancouver, Canada)

<https://artint.info/html/ArtInt_271.html>

This website was essential for us in understanding the SARSA algorithm and its implementation. Specifically, chapter 11.3.9.1 (SARSA with Linear Function Approximation) helped us better understand what each parameter in the function required, and why it was important.

1. Reinforcement Learning to solve Rubik’s Cube

<https://medium.com/datadriveninvestor/reinforcement-learning-to-solve-rubiks-cube-and-other-complex-problems-106424cf26ff>

This Medium article helped us get started with the problem formulation and helped us start thinking about how to solve the Rubik’s cube using different approaches during the early phases of our project.

# Extra Credit – Partner’s Reflections

## Reflection 1: Michelle Ho

In this project, I worked through the problem formulation, feature representation of states, and the SARSA algorithm. I helped verify Godwin’s work, debug, and optimize the code.

Throughout this project, having Godwin as a partner helped tremendously with debugging, optimizing code, and bouncing ideas on design decisions. There were no challenges with working with Godwin on this project, as he was a very proactive partner with many ideas on how to continue to optimize our algorithm.

## Reflection 2: Godwin Vincent