Solving a Rubik's Cube

Using Reinforcement Learning

Todd Graham, Vince Olsen, William McIntosh, and William Papsco

Abstract

Using reinforcement learning tabula rasa to solve a Rubik's cube is a difficult task due to the large state space and single goal state, which is usually considered to be the only state in which the agent receives a reward. In this project, we supplemented the learning with an approach called autodidactic iteration (ADI). Reinforcement learning is an area of machine learning concerned with how intelligent agents ought to take actions in an environment in order to maximize the notion of cumulative reward (Russell). We solved this by using ADI which is a learning algorithm that is able to teach itself how to solve the Rubik's cube with no human assistance. Using ADI, our agent was able to far surpass the naïve agent, it was approximately 6 times more likely to solve the cube from scramble depths 4 to 12.

Introduction

Reinforcement learning seeks to create intelligent agents that adapt to an environment by analyzing their own experiences, learning from success and failure, from reward and punishment. For example, in chess, the agent needs to know that something good has happened when it (accidentally) checkmates the opponent, and that something bad has happened when it is checkmated. This kind of feedback is called a reward, or reinforcement. A

Q-learning agent learns an action-utility function, or Q-function, giving the expected utility of taking a given action in a given state (Russell 830).

Related Work

Youness Mansar’s article titled “Learning To Solve a Rubik’s Cube From Scratch using Reinforcement Learning” concerns the same reinforcement learning approach implemented also using ADI (Mansar). Youness also constructs a path to the solved state by going backwards from the solved state. The reward is the only supervision signal that the network gets during training, which can be delayed by multiple “action” decisions since the only state with a positive reward is the final solved state.

Rubik's Cube

History

The Rubik's cube is a 3-D combination puzzle invented in 1974 by Hungarian sculptor and professor of architecture Ernő Rubik. The puzzle was licensed by Rubik to be sold by Ideal Toy Corp. in 1980 via businessman Tibor Laczi and Seven Towns founder Tom Kremer (Fotheringham). Rubik's cube won the 1980 German Game of the Year special award for Best Puzzle. As of January 2009, 350 million cubes had been sold worldwide, making it the world's top-selling puzzle game (Adams).

Overview

The Rubik's cube consists of 26 smaller cubes called cubelets. There are six central pieces that show one colored face and each face has 3x3 or 9 cubelets (Adams). There are twelve edge pieces which show two colored faces, and eight corner pieces which show three colored faces. Each piece shows a unique color combination. The location of these cubes relative to one another can be altered by twisting an outer third of the cube by increments of 90 degrees, but the location of the coloured sides relative to one another in the completed state of the puzzle cannot be altered; it is fixed by the relative positions of the center squares. An internal pivot mechanism enables each face to turn independently, thus mixing up the colors. For the puzzle to be solved, each face must be returned to have only one color.

Complexity Analysis

The original (3×3×3) Rubik's cube has eight corners and twelve edges. There are $8!$ (40,320) ways to arrange the corner cubes. Each corner has three possible orientations, although only seven (of eight) can be oriented independently; the orientation of the eighth (final) corner depends on the preceding seven, giving 3^7 (2,187) possibilities (Holt). There are $\frac{12!}{2}$ (239,500,800) ways to arrange the edges, restricted from 12! because edges must be in an even permutation exactly when the corners are. When arrangements of centers are also permitted, as described below, the rule is that the combined arrangement of corners, edges, and centers must be an even permutation. Eleven edges can be flipped independently, with the flip of the twelfth depending on the preceding ones, giving 2^{11} (2,048) possibilities (Schönert).

$$8! \times 3^7 \times \frac{12!}{2} \times 2^{11} = 43,252,003,274,489,856,000$$

Human Solving Methods

While there are many different ways to solve a Rubik's cube, most people start with what is known as the "beginner's method." This method begins by creating a white cross on the bottom face of the cube. It's important to ensure that the edge pieces used for the cross also match up on the other sides. Many people think of the Rubik's cube as a puzzle about getting all 54 stickers in the right spots, but really it's about getting the 8 corner pieces and 12 edge pieces to the right positions and in the correct orientation. For example, if the solver makes a white cross but none of the stickers on the side match, none of the pieces are actually in the correct position.

Once the cross is made, the next step in the beginner's method is to solve the bottom two layers of the cube. This is done by first inserting the corner pieces with white stickers into the bottom face and then inserting the correct edge pieces. Up until this point, most of these steps can be completed through intuition. Solving the last layer is when intuition ceases to be helpful. Learners must memorize four algorithms, or sequences of moves, to solve the last layer. The first algorithm re-orients the top edge pieces until all 4 edge pieces are in the correct orientation. The second algorithm swaps two top edge pieces and can be used to get the edges to the correct position. Now, the top edge pieces are solved and we must now solve the corner pieces. As with the edge pieces, there are two algorithms for solving the corner pieces: one to orient them and one to position them. Once the corners are solved, the whole cube should also be solved.

More advanced solving methods involve consolidating steps by learning additional algorithms to save time. Instead of inserting corners and then edges to solve the first two layers, advanced solvers pair edges and corners together before inserting them together. To solve the

last layer, a two-step process can be learned by memorizing algorithms for each permutation of the last layer. First, all pieces in the last layer are oriented correctly such that the same color faces up in one algorithm. Next, the pieces are positioned correctly in one additional algorithm. Whereas the beginner's method requires 4 steps with multiple algorithm usages in each step, the advanced method allows the last layer to be solved in only two algorithms.

Reinforcement Learning

Base Method

In this project we decided to use the classic reinforcement learning approach of Q-Learning. To avoid the need for training data, a Q-Table was used in lieu of creating a deep Q learning model. Keys in the Q-Table were states of the cube with an array of the 12 possible moves as values for each key. Due to the massive size of the state space the Q-Table was not pre-initialized. As the agent progressed through the learning process, if it encountered a state that was not in the Q-Table, it was added with values of 0 for each of the possible actions.

Autodidactic Iteration (ADI)

Reinforcement learning works by rewarding good choices and not rewarding or negatively rewarding bad choices. In most applications, the reward is issued at the end of each turn, such as how many points were earned. When solving a Rubik's cube, there is only one goal state, so there is only one reward at the end of the puzzle. In the absence of rewards, the learning algorithm makes random choices. The state space is very large and there is no guarantee that the puzzle will be solved with random actions. This can make a reinforcement

learning algorithm very inefficient at solving a Rubik's cube. The problem of sparse reinforcement is overcome by taking the binary reward of the goal state and smearing it out over the states that lead to the goal state, essentially creating a funnel of ever increasing rewards that leads the agent to the goal state. This turns the very small target of the goal space into a very large target that is easier for the learning agent to find.

We achieved this "smeared-out" reward space by implementing a parameterized ADI. We started at the goal state with a depth parameter and a reward coefficient parameter. We then worked backward from the goal state using the available actions to find the successor states. We applied a reward to each state and action combination that was weighted according to the proximity to the goal state and stored these values in a dictionary for quick look up. The reward was positive if the action took you from the current state to the next state closer to the goal, and the reward was negative otherwise. ADI made our reinforcement learning agent approximately 6 times more efficient at solving the Rubik's cube from initial scramble depths of 4 to 12..

Methods

Using Q-Learning in combination with ADI, our goal was to tune the hyperparameters eta (learning rate) and gamma (discount factor) to test how far from the goal state our agent could solve after training. To optimize our hyperparameters, we tested 0.1 increments of both eta and gamma in the range [0, 0.9]. Starting by optimizing eta, we held gamma, the ADI depth and the number of shuffles for the test cubes constant. The Q-Table was pre-trained using ADI to a depth of 4 from the goal state, and the starting state of both the training and test cubes were 6 moves away from the goal state. The ADI training depth of 4 was chosen to balance computational efficiency and effectiveness. The agent was trained over 5000 epochs then

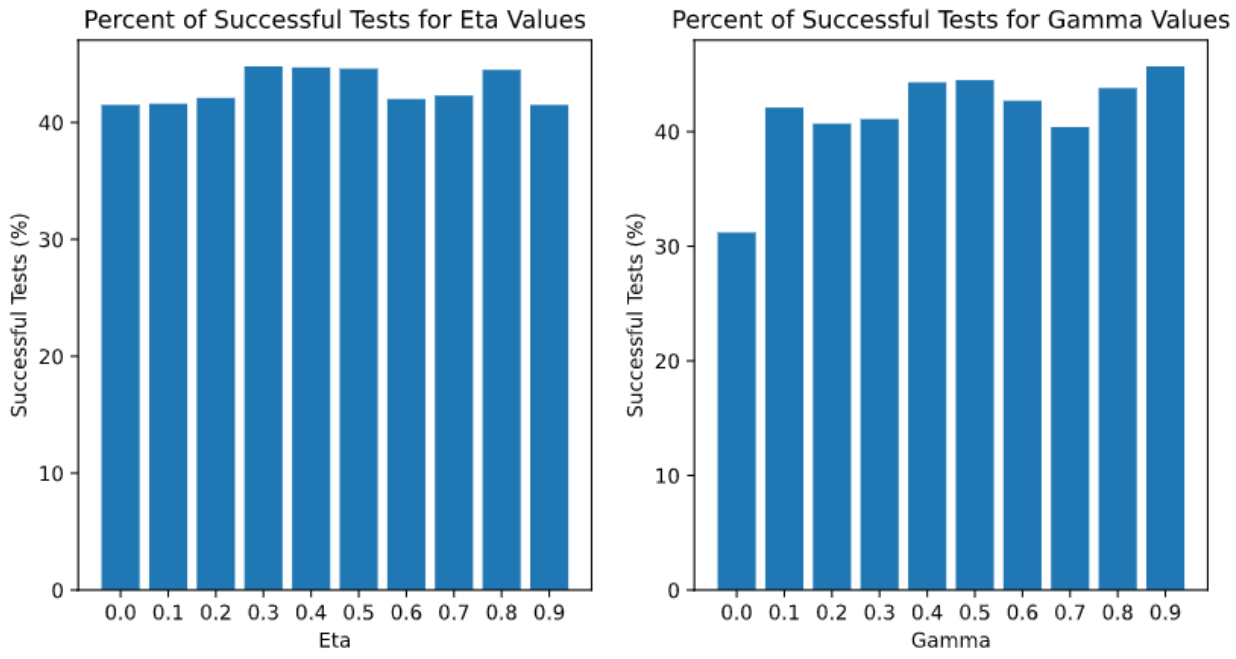
tested with 1000 test cubes. The eta value that resulted in the most solved cubes was then used to test the 10 gamma values.

Once the hyperparameters were chosen, we wanted to test how many moves away from the ADI depth the agent would be able to solve the cube. To do this, we trained the agent over 5000 epochs then tested with 1000 test cubes for cubes that were between 0 and 9 moves from the ADI depth.

Results

An eta value of 0.3 resulted in a 44.8% success rate for tests which was the highest among the values tested. The eta values 0.4, 0.5 and 0.8 were the next closest with success rates of 44.7%, 44.6% and 44.5% respectively. The eta values with the lowest success rates were 0 and 0.9 with a success rate of 41.5%. The full results of this experiment can be found in Figure 1.

A gamma value of 0.9 resulted in a 45.7% success rate for tests which was the highest among the values tested. The gamma values of 0.5, 0.4, 0.8 were the next closest with success rates of 44.5%, 44.3% and 43.8% respectively. The gamma value with the lowest success rate was 0, with a success rate of 31.2%. The full results of this experiment can be found in Figure 2.



Figures 1 & 2. Hyperparameter optimization. Successful test percentage after 5000 training epochs.

After hyperparameter tuning, we tested the agent's ability to solve cubes that had been scrambled various amounts. Results of agent's with ADI are shown in Figure 3 and without ADI are shown in Figure 4. As expected in Figure 3, 100% of cubes were solved when they were at the level of pre-training, in comparison to 16.6% without pre-training. As both experiments reached a depth of 9 past the pre-training depth, both agents performed very poorly, with 0.4% and 0.3% of test cubes being solved.

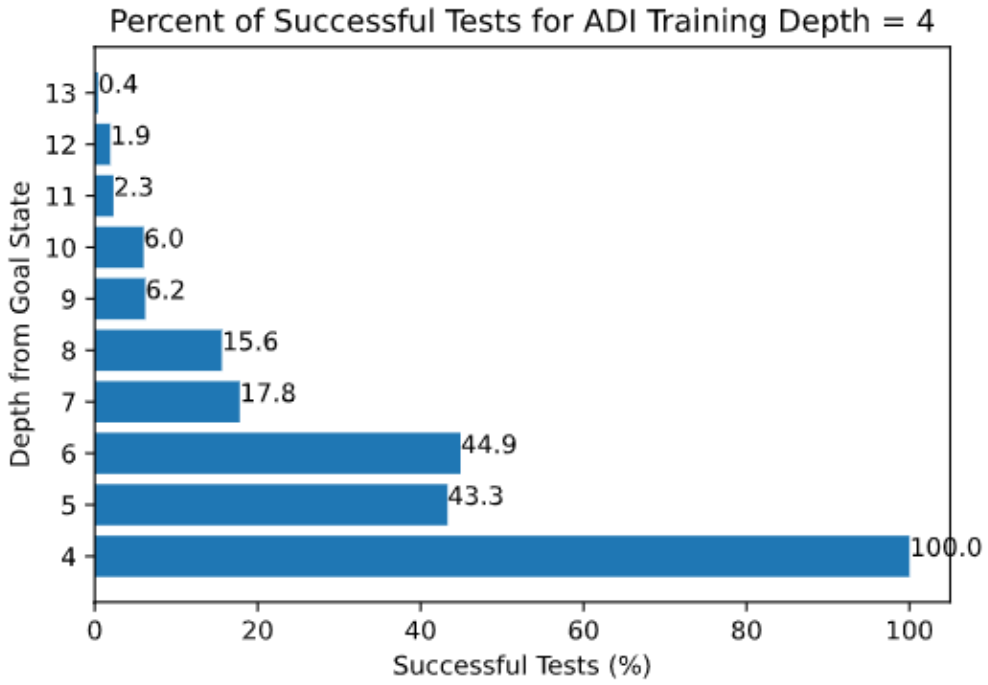


Figure 3. Agent's performance with varying depths from the ADI depth.

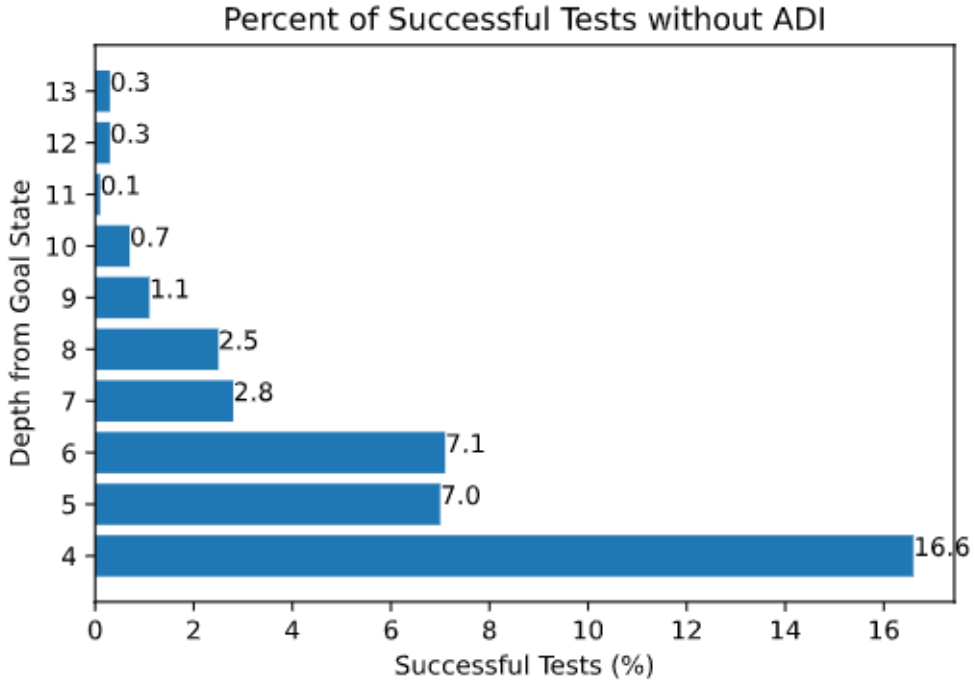


Figure 4. Agent's performance with varying depths without ADI.

Conclusions

We optimized a reinforcement learning algorithm with eta = 0.3 and gamma = 0.9 and used autodidactic iteration to solve a Rubik's cube. Using ADI with the algorithm made it approximately 6 times more likely to solve the puzzle than without ADI. These results surprised us and surpassed both the results of the preceding paper as well as the Rubik's cube skillz of our team members.

References

Course Group Project Repo Link

- https://github.com/toddgraham121/cs541-rubik-cube-solver

Bibliography

Adams, William Lee. *The Rubik's Cube: A Puzzling Success*. Time, 2009.

Fotheringham, William. *Fotheringham's Sporting Pastimes*. ISBN 978-1-86105-953-6 ed., Anova Books, 2007.

Holt, K. "AI learns to solve a Rubik's Cube in 1.2 seconds." 2019, https://www.engadget.com/2019-07-17-ai-rubiks-cube-machine-learning-neural-network.html.

Mansar, Youness. "Learning To Solve a Rubik's Cube From Scratch using Reinforcement Learning." 2019,

https://towardsdatascience.com/learning-to-solve-a-rubiks-cube-from-scratch-using-reinforcement-learning-381c3bac5476.

Russell, Stuart J. *Artificial Intelligence A Modern Approach*. Third Edition ed., vol. ISBN-13: 978-0-13-604259-4, Pearson Education, Inc., 2010.

Schönert, Martin. “Analyzing Rubik's Cube with GAP.” *Analyzing Rubik's Cube with GAP*, 1993, https://www.gap-system.org/Doc/Examples/rubik.html.