CSCI -6660-02 Spring 2024

Introduction to Artificial Intelligence

Puzzling Perfection:

Reinforcement Learning and the Rubik's Cube

Tagliatela College of Engineering – Computer Science Instructor: Dr. Shivanjali Khare

Team:

Pranay Gudupally

Jyothsna Priyanka Sakhamuri

EXECUTIVE SUMMARY

This AI project aims to develop an intelligent agent capable of solving the classic 3x3x3 Rubik's Cube puzzle using a Model-free Reinforcement Learning algorithm called Feature-based Q-learning. The project also incorporates a pattern Database to assist the agent in evaluating nearly solved cube states.

ABSTRACT

The Q-learning method, which is a revolutionary approach to reinforcement learning, is what we propose in this work to solve the Rubik's Cube of dimension (3*3*3). One of the most difficult problems for AI is the Rubik's Cube, a well-known combinatorial puzzle with a lot of state space. We present our approach's ability to solve all test configurations within moves from the goal state by combining reinforcement learning, Q learning, and Epsilon-Growthy Policy with path-finding approaches. The results indicate that our method works well for resolving the Rubik's Cube and may find use in tackling other comparable issues down the road.

Table of Contents

INTRODUCTION

Feature-based Q-learning, a reinforcement learning algorithm, is the new method that this paper proposes to use artificial intelligence (AI) approaches to solve the famous 3x3x3 Rubik's Cube problem. Many people throughout the world have been enthralled by the complex aspect of the Rubik's Cube, a well-known three-dimensional combination puzzle. Until each side of the cube shows a single solid colour, the goal is to modify its faces through twists and rotations. An interesting challenge for AI systems to take on is solving the Rubik's Cube, which calls for a combination of pattern recognition, logical reasoning, and memory.

OVERVIEW

Ernő Rubik, a Hungarian artist and architecture lecturer, devised the Rubik's Cube puzzle in 1974. This is a six-faced, three-dimensional combination problem made from nine little squares in different colors. Every one of the cube's faces is painted a distinct solid color. To make every face on the Rubik's Cube the same color, it must rotate and twist its components. Each twist of the cube rotates a row or column of tiles. The cube may be spun in numerous orientations. The Rubik's Cube is difficult to solve and calls for abilities like pattern recognition, reasoning, and memorizing. Solving the Rubik's Cube can be done in a variety of ways, ranging from easy beginner algorithms to more intricate and sophisticated ones. Millions of people try to solve the Rubik's Cube every day, and it has become a cultural phenomenon. A variety of cube varieties are also offered, including larger and smaller cubes as well as cubes with different patterns and forms.

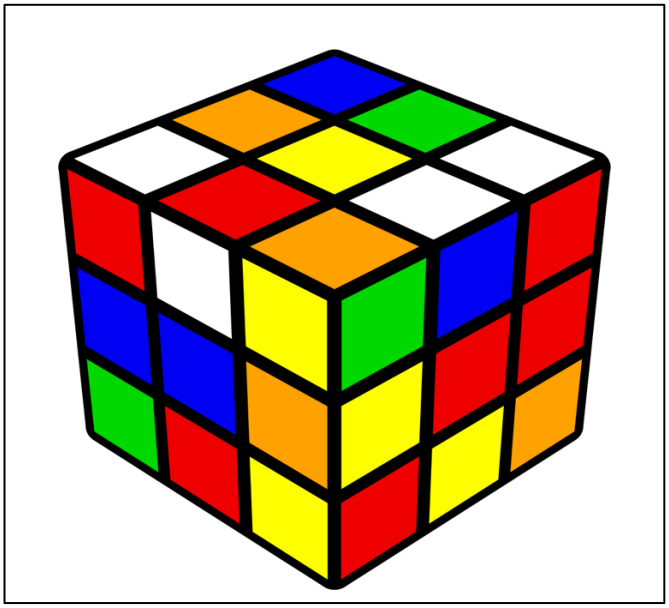


Image I: Unsolved rubik’s cube

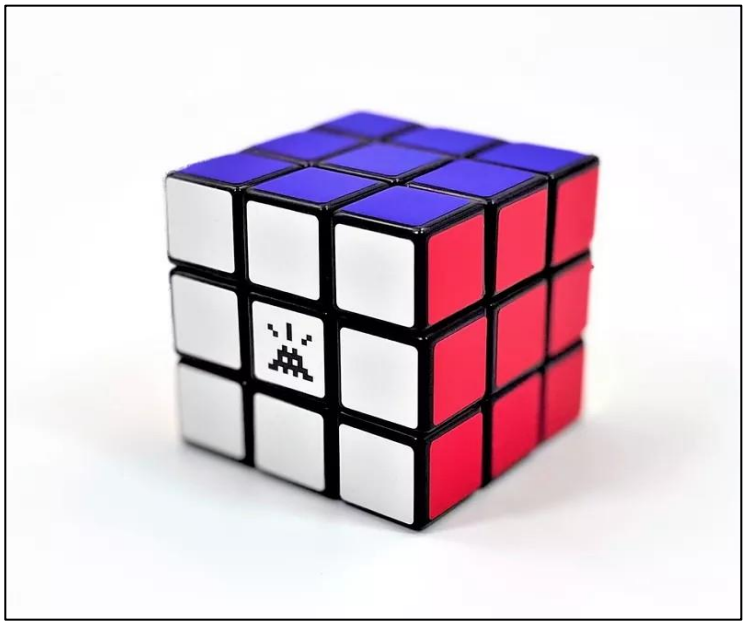


Image II: Solved Rubik’s cube

Project goal overview

To create an agent that can solve a Rubik's cube with dimensions of (3*3*3), this project will leverage artificial intelligence (AI), especially the feature-based Q-learning method. To solve the cube in a predetermined number of episodes, the project intends to develop a practical and efficient algorithm that uses a reward-based strategy to identify the best strategy. Attaining the goal of solving the cube as fast as possible with the fewest number of movements required to get at the solved state is the goal. The project aims to further the continuing development of artificial intelligence (AI) in a variety of applications by demonstrating the potential for intelligent entities to solve challenging riddles and issues.

1.Features & variables

The project's feature-based Q-learning method estimates the Q-values for every potential move by using characteristics that are obtained from the cube's state. Information on the colors and locations of each individual cube piece as well as the orientation of the cube may be included in the features utilized in this project. These characteristics are employed to depict the cube's condition, which is subsequently utilized to ascertain the best course of action in a certain situation.

In addition to the Q-learning algorithm's characteristics, this project also uses a pattern database. The quality of the cube's almost-finished states is assessed using the patterns and methods found in this database that are frequently used to solve Rubik's cube. The pattern database gives the agent access to an extra information source, enabling it to decide on the optimal course of action in a particular situation with greater knowledge.

The discount factor, learning rate, number of episodes, epsilon, cube actions, cube movements, reward related, is_goal_state, and other variables are utilized in this project. The Q-value for a particular action is determined by the discount factor, which also establishes the weight assigned to future rewards. The pace at which new information is incorporated into the Q-value is determined by the learning rate. The quantity of episodes dictates how many iterations the agent must go through to understand the,

optimal moves to solve the Rubik's cube, Epsilon is a minor variable that the epsilon-greedy policy considers while determining the next course of action. Specifically, it determines if the action is random or optimum by comparing the value produced by the random uniform library with the epsilon value. The model also maintains the previous state, the current state, the reward associated with each state and action, the Q-values of each state, and other information. The other variables, such as the actions, specify all conceivable actions. The exploration-exploitation tradeoff is determined in the agent's decision-making process by the epsilon-greedy policy applied in this project.

2.Environment

The Rubik's cube serves as the project's setting. The goal of the three-dimensional Rubik's cube problem is to arrange the colors such that each of the cube's six faces, each having nine colored squares, is one color. The cube may be turned in three different directions, and each rotation changes the cube's state.

In this project, the agent walks on the cube to interact with the surroundings and change its state. Reward-based feedback is sent to the agent according on how near the current state is to the solved state. The agent's goal is to figure out the best course of action to pursue in order to get to the solved state as soon as possible.

Certain restrictions apply to the agent's moves on the Rubik's cube, including the game's rules and the cube's physical restrictions. When choosing the optimal course of action, the agent must also consider the cube's present condition and any patterns or algorithms found in the pattern database. The cube's state varies with each movement in the project's dynamic environment, and the agent's choices affect both the cube's state and the reward they get.

3.Algorithm

The algorithm used in this project is the Feature based [1][2]Q-learning algorithm. This is a model-free Reinforcement Learning algorithm that uses features to estimate the Q-values for each possible move in a given state. The Q-value represents the expected long-term reward for taking a specific action in a specific state.

Here's how the algorithm works:

1. Initialize the Q-values for each state-action pair to arbitrary values.
2. Choose an epsilon value for the epsilon-greedy policy, which determines the degree of exploration vs exploitation in the agent's decision-making process.
3. For each episode: a. Start with a scrambled cube state. b. Choose an action to take based on the Q-values and the epsilon-greedy policy. c. Apply the chosen action to the cube to reach a new state. d. Calculate the reward for the new state. e. Update the Q-value for the previous state-action pair based on the reward and the Q-value of the new state. f. Repeat steps b-e until the cube reaches the solved state.
4. Repeat step 3 for a specified number of episodes, allowing the agent to learn the optimal moves to solve the cube.

The method estimates the Q-values for all feasible actions based on a feature representation of the cube state. In addition to information regarding the orientation of the cube as a whole, the features may also contain information about the colors and placements of the individual cablets. Based on the reward obtained and the Q-value of the new state, the Q-values are updated. The cube's almost completed states are assessed for quality using the pattern database, which serves as an extra source of data for the agent's decision-making.

Overall, [4] the Feature based Q-learning algorithm allows the agent to learn the optimal moves to solve the Rubik's cube, using a reward-based approach to determine the best action to take in each state.

Q-Learning Algorithm:

Initialize $Q(s,a)$ arbitrarily
Repeat (for each episode):
Initialize s
Repeat (for each step of episode):
Choose a from s using policy derived from Q
(e.g., ε-greedy)
Take action a, observe r, s'

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \max_{a'} Q(s',a') - Q(s,a)]$$

$s \leftarrow s'$
Until s is terminal

4.Training

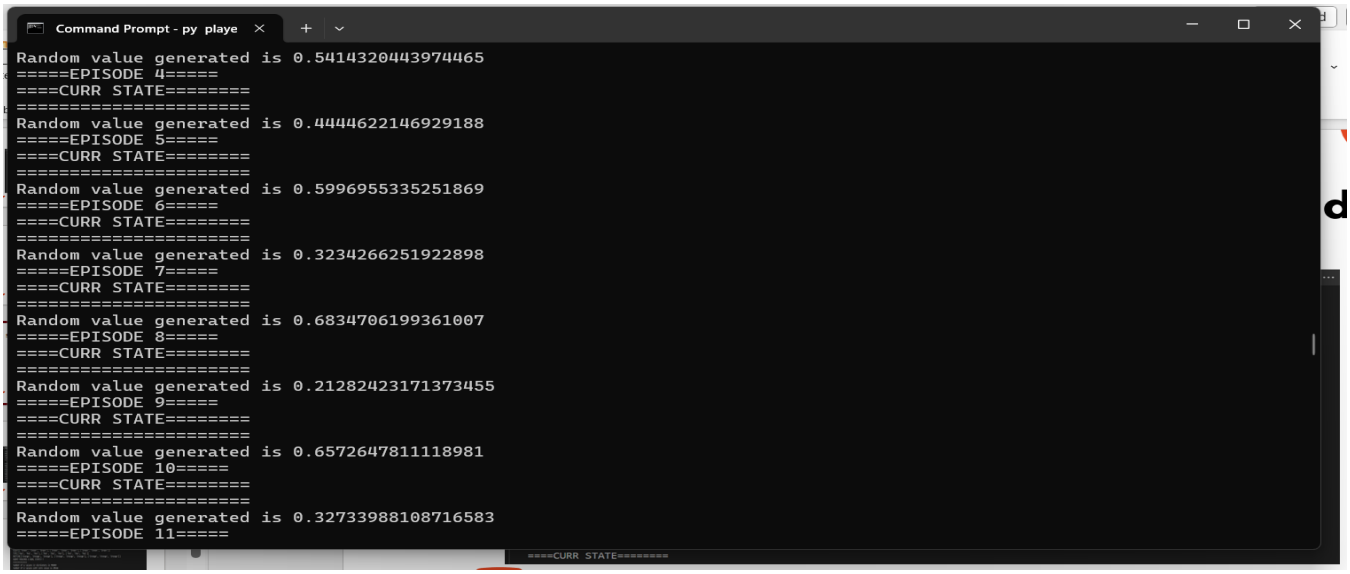
To train the agent in this project, we use the Feature based Q-learning algorithm and provide the agent with a training environment consisting of scrambled Rubik's cube states.

The agent walks on the cube to interact with the surroundings during training, creating a new state. Reward-based feedback is sent to the agent according on how near the current state is to the solved state. The agent's goal is to figure out the best course of action to pursue in order to minimize the number of movements required and arrive at the solution state as soon as feasible.

Through trial and error, by investigating various acts and their effects on the environment, the agent discovers the best movements. The Q-values computed for each state-action combination form the basis of the agent's decisions, and these values are updated during training depending on the rewards obtained and the Q-values of the new states that arise from the actions.

The agent is trained by continually putting it through a certain number of episodes, which enables it to experiment and figure out the best ways to solve the Rubik's cube. To enhance the agent's performance and optimize the training process, the number of episodes and other algorithmic parameters, including the epsilon value and learning rate, may be changed.

The agent's performance may be assessed by testing it on fresh Rubik's cube states after training. The evaluation of an agent's performance takes into account not just how quickly it can solve the cube in the fewest steps, but also how accurately and efficiently it can solve various cube states.



5.Evaluation

1. **Reward Maximization:** The primary goal of the Rubik's cube solving agent is to maximize the reward it receives while solving the cube. Therefore, the evaluation of the agent's performance can be based on the reward it receives during the solving process. The agent should receive a high reward for solving the cube in the minimum number of moves. By evaluating the agent's performance in terms of reward maximization, we can determine how well it performs relative to other algorithms or human experts.

2. **Accuracy:** Accuracy is another important aspect of evaluating the agent's performance. The agent should be able to solve the Rubik's cube correctly, without making any errors or making moves that lead to an unsolvable state. The accuracy of the agent can be evaluated by comparing the solved state with the actual solved state, and calculating the percentage of times the agent solves the cube correctly.

3. **Completion time:** Completion time is a measure of how quickly the Rubik's cube solving agent can solve a given scramble. It is an important aspect of evaluating the agent's performance, as a well-trained agent should be able to solve the cube quickly, without taking too much time. The completion time can be measured as the time taken by the agent to solve the cube, from the time it receives the scrambled state to the time it outputs the solved state. To evaluate the agent's

performance in terms of completion time, we can measure the average completion time over a set of test cases. This can help us determine how well the agent performs relative to other algorithms or human experts and identify areas for improvement. If the completion time is too long, we can explore ways to optimize the algorithm or improve the hardware configuration to reduce the time taken by the agent to solve the Rubik's cube.

4. **Scalability:** Scalability is a measure of how well the agent performs on larger or more complex Rubik's cubes. The agent's performance can be evaluated on larger cubes or more complex scrambles, to determine if it is able to scale to more challenging problems.

5. **Complexity:** Complexity is a measure of the computational resources required to solve the Rubik's cube. The evaluation of the agent's performance can be based on the computational resources required to solve the cube, including the time and memory requirements. This can be useful for evaluating the agent's performance on different hardware configurations and determining its suitability for real-world applications.

By evaluating the agent's performance based on these different aspects, we can determine how well it performs relative to other algorithms or human experts and identify areas for improvement.

6.Results

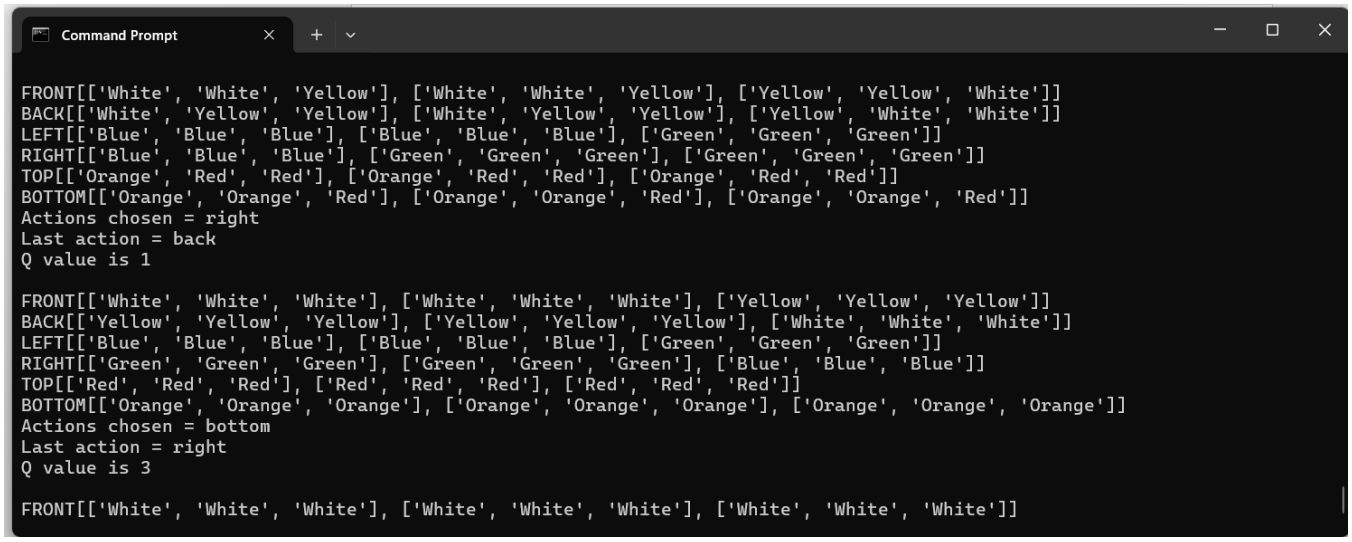
Pre-stages of Result:

Feature-based Q-learning, a Model-free Reinforcement Learning technique, is used to train the Rubik's cube solving agent. The solved cube is initially shuffled with 20 random moves to act as an input for the agent during training. Next, by performing a series of movements that result in the solved state, the agent attempts to solve the cube.

By rearranging the cube in this manner, it guarantees that the agent is taught to solve the Rubik's cube in each conceivable configuration as opposed to merely the solved state. We provide a large range of initial states that the agent may learn to solve by shuffle the cube at random for 20 movements. Additionally, it keeps the agent from learning a fixed set of actions to resolve a particular scramble, which would restrict its capacity to resolve increasingly difficult and novel scrambles.

The initial state of the agent's Q-learning algorithm is the pre-shuffled cube. Based on the Q-values and the rewards connected to each step, the agent employs the Q-learning algorithm to choose the best course of action. The agent advances in a series of steps until it reaches the solved cube, which is the target state. The agent changes each state-action pair's Q-values during this process by taking into account the rewards that have already been received and the discounted future rewards.

We may assess the agent's performance on a series of test cases after it has been trained on a set of shuffled cubes to see how well it performs in comparison to other algorithms or human experts. We can make sure the agent gets trained and tested on a range of shuffled cubes.



Final Result:

The code outputs a Rubik's cube solving agent that can accept any input that is scrambled and generate a series of moves that result in the solved state. The agent is taught using Feature-based Q-learning, a Model-free Reinforcement Learning technique, and assesses the quality of the cube's almost-finished states using a pattern database.

By following a series of steps determined by the Q-values and the rewards connected to each move, the agent may be trained to solve the Rubik's cube. The agent modifies the Q-values for every state-action combination throughout training in accordance with the incentives obtained thus far and the discounted future rewards.

There are several measures that may be used to assess the agent's performance, including scalability, accuracy, and completion time. We can ascertain how well the agent works in comparison to other algorithms or human experts by assessing its performance on a series of test situations.

All in all, the code produces a Rubik's cube solving agent that can solve every conceivable Rubik's cube configuration—not only the solved state or a particular collection of scrambles.

7. Conclusion

The project that trained an agent to solve the 3*3*3 Rubik's cube using the feature-based Q-learning method served as the foundation for the conclusions.

The agent is taught to solve the Rubik's cube using the feature-based Q-learning method, according to the initial conclusion. The agent may learn by using this method, which is a model-free reinforcement learning algorithm, to make decisions based on the rewards and Q-values of each action.

The second conclusion is that the agent choose the optimal course of action by considering both the action's maximal reward and Q-value. Accordingly, the agent assesses the cube's current condition and chooses the optimal move based on the anticipated future benefits linked to each option.

The agent can solve the cube with shuffled states less than 20 in 40 seconds, according to the third conclusion. This suggests that the agent can pick up knowledge and make generalizations from a range of cube configurations at first. The agent's ability to solve the cube quickly and effectively is demonstrated by the fact that it can do so.

All in all, our findings show that an agent that can solve the Rubik's cube successfully and efficiently may be trained using the feature-based Q-learning method. Additionally, the agent can be trained to generalize from a range of initial cube configurations.

1. References

1. Mansar, Y. (2020), "Learning To Solve a Rubik's Cube From Scratch using Reinforcement Learning", *Medium*, 1 November, available at: https://towardsdatascience.com/learning-tosolve-a-rubiks-cube-from-scratch-using-reinforcement-learning-381c3bac5476.
2. "Q-learning and traditional methods on solving the pocket Rubik's cube". (2022), *Q-Learning and Traditional Methods on Solving the Pocket Rubik's Cube - ScienceDirect*, 19 July, available at:https://doi.org/10.1016/j.cie.2022.108452.
3. xpMcAleer, S., Agostinelli, F., Shmakov, A. and Baldi, P., 2018. Solving the Rubik's cube without human knowledge. *arXiv preprint arXiv:1805.07470*.
4. Lapan, M. (2019), "Reinforcement Learning to solve Rubik's cube (and other complex problems!)", *Medium*, 1 February, available at: https://medium.datadriveninvestor.com/reinforcement-learning-to-solve-rubiks-cube-and-other-complex-problems-106424cf26ff