

CSCI -6660-01 Intro Artificial Intelligence Fall 2023

**Feature Based Reinforcement Learning for The Rubik’s Cube(4\*4\*4)**

**Team Members**

**Manasa Sukavasi**

**Indhu Allala**

**Sai Chandan Regati**

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

#### ABSTRACT

In this research, we present an innovative strategy for solving the Rubik's Cube of dimension (4\*4) using reinforcement learning, specifically the Q-learning algorithm. The Rubik's Cube is a well-known puzzle with a vast number of possible combinations, making it a difficult challenge for artificial intelligence. We showcase how our approach, which combines Q-learning, Epsilon-greedy Policy, and path-finding techniques through reinforcement learning, can successfully solve all test configurations within a few moves from the desired end state. Our results indicate that our method is effective in tackling the Rubik's Cube and has the potential to be applied to similar problems in the future.

Table of Contents

1. [Introduction 4](#_TOC_250009)
2. [Overview 4](#_TOC_250008)
   1. Rubik cube overview 4
   2. Project goal overview 6
3. [Features & variables 6](#_TOC_250007)
4. [Environment 7](#_TOC_250006)
5. [Algorithm 8](#_TOC_250005)
6. [Training 9](#_TOC_250004)
7. [Evaluation 11](#_TOC_250003)
8. [Results 12](#_TOC_250002)
9. [Conclusion 15](#_TOC_250001)
10. [References 16](#_TOC_250000)

# **Introduction**

In this research, we present an innovative strategy to solve the Rubik's Cube of dimension (4\*4) using reinforcement learning, specifically the Q-learning algorithm. The Rubik's Cube is a well-known puzzle with numerous possible combinations, posing a significant challenge for artificial intelligence. Through our study, we demonstrate the effectiveness of our approach, which combines Q-learning, Epsilon-greedy Policy, and path-finding techniques, in solving all test configurations within a few moves from the desired state. These findings imply that our method holds promise for successfully solving the Rubik's Cube and potentially addressing similar problems in the future.

* 1. ***Rubik cube overview***

# **Overview**

The Rubik's Cube, invented in 1974 by Ernő Rubik, a Hungarian sculptor and architecture professor, is a 3D puzzle consisting of six faces with nine small colored squares on each face. The cube's faces are painted in different solid colors, and the goal is to manipulate and rotate the cube's parts to achieve uniform colors on all faces. The cube can be turned in multiple directions, with each twist rotating an entire row or column of squares. Solving the Rubik's Cube requires skills such as pattern recognition, logical thinking, and memorization. There are various techniques and methods available to solve the cube, ranging from simple beginner algorithms to more advanced ones. The Rubik's Cube has gained immense popularity worldwide, with millions of people attempting to solve it. Additionally, there are different variations of the cube available, including larger and smaller sizes, as well as cubes with unique patterns and shapes.

A stack of multicolored cubes

Description automatically generated

**Image I:** Unsolved rubik’s cube

A rubik's cube on a table

Description automatically generated

**Image II:** Solved Rubik’s cube

* 1. ***Project goal overview***

The objective of this endeavor is to utilize artificial intelligence (AI), specifically the Feature based Q-learning algorithm and pattern Database, to construct an agent capable of solving a Rubik's cube with dimensions of 4x4. The main goal is to develop an algorithm that is both efficient and effective, enabling the agent to solve the cube within a predetermined number of attempts by employing a reward-based approach to determine the optimal moves. The ultimate aim is to create an agent that can swiftly solve the cube while minimizing the number of moves required to reach the solved state. By employing AI techniques, this project seeks to showcase the potential of intelligent agents in solving intricate puzzles and problems, thereby contributing to the continuous advancement of AI across various domains.

# **Features & variables**

The Feature-based Q-learning algorithm employed in this project is founded on the use of features, which are obtained from the current state of the cube, to estimate the Q-values for each possible move. The features utilized in this endeavor can consist of details about the colors and positions of the individual cubelets, as well as information regarding the orientation of the cube as a whole. These features serve to represent the state of the cube, which is then utilized to ascertain the optimal move in any given state.

Alongside the features employed in the Q-learning algorithm, this project also incorporates a pattern database. This database encompasses information about common patterns and algorithms employed to solve the Rubik's cube. These patterns and algorithms are subsequently used to evaluate the quality of nearly completed states of the cube. The pattern database serves as an additional source of information for the agent, aiding it in making more informed decisions regarding the best move to execute in a particular state.

The variables utilized in this project encompass the discount factor, learning rate, number of episodes, epsilon, cube actions, cube moves, reward associated with each move, and whether or not the goal state has been reached. The discount factor determines the significance assigned to future rewards

# **Environment**

The focal point of this project revolves around a three-dimensional puzzle known as the Rubik's cube. With its six sides, each adorned with nine colored squares, the objective is to arrange the colors in a manner that each side of the cube displays only one color. By rotating the cube along three axes, players have the ability to create various configurations. Within this project, an agent interacts with the Rubik's cube by executing moves, resulting in a new state. Feedback in the form of rewards is provided to the agent, based on how closely the current state resembles the solved state. The primary objective of the agent is to learn the most effective moves that will lead to the solved state as quickly as possible, while minimizing the total number of moves made.

The actions executed by the agent on the Rubik's cube are subject to specific limitations, including the physical constraints of the cube's movements and the rules of the game. Additionally, when determining the optimal move to make, the agent must take into account factors such as the current state of the cube and any patterns or algorithms stored in the pattern database. The environment of this project is dynamic, with the state of the cube changing after each move and the agent's decisions being influenced by these changes.

# **Algorithm**

The algorithm employed in this project is the Feature-based Q-learning algorithm [1][2]. This particular algorithm falls under the category of model-free Reinforcement Learning algorithms, and it utilizes features to approximate the Q-values for each possible move within a given state. The Q-value, in this context, signifies the anticipated long-term reward for executing a specific action in a specific state.

Here is a breakdown of how the algorithm operates:

1. Initialize the Q-values for every state-action pair with arbitrary values.
2. Determine an epsilon value for the epsilon-greedy policy, which governs the balance between exploration and exploitation in the decision-making process of the agent.
3. For each episode: a. Begin with a scrambled configuration of the cube. b. Decide on an action to take based on the Q-values and the epsilon-greedy policy. c. Apply the chosen action to the cube, resulting in a new state. d. Compute 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 and determine the optimal moves required to solve the cube.

To estimate the Q-values, the algorithm employs a feature representation of the cube's state.

A white background with black text

Description automatically generated

# **Training**

To train the agent in this particular project, we utilize the Feature based Q-learning algorithm. We provide the agent with a training environment that consists of Rubik's cube states that have been scrambled.

Throughout the training process, the agent interacts with the environment by making moves on the cube. Each move results in a new state. The agent receives feedback in the form of a reward, which is determined by how close the current state is to the solved state. The main objective of the agent is to learn the optimal moves that will lead to reaching the solved state as quickly as possible, while minimizing the number of moves required.

The agent acquires knowledge of the optimal moves through trial and error. It explores different actions and their consequences within the environment. The agent's decisions are based on the Q-values estimated for each possible action in relation to a given state. These Q-values are updated during training based on the rewards received and the Q-values associated with the resulting new states resulting from the actions taken.

The training process involves repeatedly running the agent through a specified number of episodes. This allows the agent to explore and learn the optimal moves required to solve the Rubik's cube. The number of episodes and other algorithm parameters, such as the learning rate and exploration rate, can be adjusted to optimize performance.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

# **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.

1. **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.
2. **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.

***Pre-stages of Result:***

# **Results**

Rubik's cube solving agent is trained using a Model-free Reinforcement Learning algorithm called Feature-based Q-learning. To train the agent, we first shuffle the solved cube with 20 random moves, which serves as an input to the agent. The agent then tries to solve the cube by making a sequence of moves that lead to the solved state.

Shuffling the cube in this way ensures that the agent is trained to solve any possible configuration of the Rubik's cube, rather than just the solved state. By randomly shuffling the cube with 20 moves, we create a wide variety of starting states that the agent can learn to solve. This also helps prevent the agent from memorizing a fixed

set of moves to solve a specific scramble, which would limit its ability to solve new and more complex scrambles.

The pre-shuffled cube serves as the initial state for the agent's Q-learning algorithm. The agent uses the Q-learning algorithm to decide the optimal move based on the Q- values and the reward associated with each move. The agent makes a sequence of moves until it reaches the goal state (solved cube). During this process, the agent updates the Q-values for each state-action pair based on the rewards received and the discounted future rewards.

After training the agent on a set of shuffled cubes, we can evaluate its performance on a set of test cases to determine how well it performs relative to other algorithms or human experts. By training and evaluating the agent on a variety of shuffled cubes, we can ensure that it is able to solve any possible configuration of the Rubik's cube, and not just the solved state or a specific set of scrambles.

A close-up of a white background

Description automatically generated

### Final Result:

The outcome of the code is an agent for solving a Rubik's cube, which has the capability to take in any jumbled Rubik's cube as input and generate a series of moves that will lead to the cube being solved. This agent is trained using a Model-free Reinforcement Learning algorithm known as Feature-based Q-learning, and it utilizes a pattern database to assess the quality of nearly finished states of the cube.

The agent can acquire the ability to solve the Rubik's cube by executing a sequence of moves based on the Q-values and the reward associated with each move. Throughout the training process, the agent updates the Q-values for each state-action combination based on the received rewards and the discounted future rewards.

The performance of the agent can be assessed using various metrics such as completion time, accuracy, and scalability. By evaluating how well the agent performs on a set of test cases, we can determine its effectiveness relative to other algorithms or human experts.

In conclusion, the result of the code is an agent for solving a Rubik's cube that is capable of solving any possible configuration of the cube, rather than just the solved state or a specific set of scrambles.

A screenshot of a computer

Description automatically generated

# Conclusion

The conclusions are based on the project that used the Feature-based Q-learning algorithm to train an agent to solve the Rubik's cube 4\*4.

The first conclusion is that the agent is trained to solve the Rubik's cube using the Feature-based Q-learning algorithm. This algorithm is a Model-free Reinforcement Learning algorithm that allows the agent to learn by making decisions based on the Q-values and the rewards associated with each action.

The second conclusion is that the agent chooses the best action based on the Q-value and the maximum reward associated with the action chosen. This means that the agent evaluates the state of the cube and determines the best move to make based on the expected future rewards associated with each possible move.

The third conclusion is that the agent can solve the cube with shuffled states <20 within 40 seconds. This indicates that the agent can learn and generalize from a variety of initial configurations of the cube. The fact that the agent can solve the cube within a specific time frame shows that it can solve the cube efficiently and effectively.

Overall, these conclusions demonstrate that the Feature-based Q-learning algorithm can be used to train an agent that is able to solve the Rubik's cube efficiently and effectively, and that the agent can learn to generalize from a variety of initial configurations of the cube.

# 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.