Machine Learning for Example Generation in Pure Mathematics

Sophia Cohen, Donald Conway, Jimmy Vineyard, Alex Yun, Advised by: Karan Srivastava, Prof. Jordan Ellenberg

University of Wisconsin - Madison

Overview: Applying AI to Math

In this project, we tried to learn relationships between algebraic groups using machine learning.

We first investigated the Euclidean Algorithm using neural networks and deep Monte Carlo Tree Search to learn the optimal next transform given a state. Next, we used Breadth-First Search and Q-learning to find optimal transformation decision boundaries in generating the Heisenberg Group, which are not well understood. After that, we used Q-learning to find decompositions of matrices in $SL_3(\mathbb{Z})$ using two generators, which is an open problem.

Machine Learning

For this project, we used neural networks in supervised and reinforcement learning to make predictions about matrix transformations. A neural network consists of layers of sequential transformations, including linear transforms, nonlinear activation functions, and normalization transforms.

Q-learning (reinforcement learning paradigm) has a machine play a 'game' where it walks around a state space and gets rewards at different states. In the tabular version, it learns a table estimating the value of each space is based on the reward at that space and the reward earned for every state after that space. Deep Q-learning makes these estimates using a neural network, rather than a table.

Deep Monte Carlo Tree Search is a reinforcement learning strategy using an actor-critic paradigm where the algorithm first semi-randomly expands a tree with child nodes representing future states, then trains deep value and state neural nets off subtree values and sampled visit frequency.

Euclidean Algorithm

Problem: given $s = \begin{bmatrix} a \\ b \end{bmatrix} \in \mathbb{Z}^2$, try to make a or b zero as fast as possible by using the following 4 transformations:

$$A = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}, B = \begin{bmatrix} 1 & -1 \\ 0 & 1 \end{bmatrix}, C = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}, D = \begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix}$$

We trained several simple neural networks on this problem. All neural networks with more than 2 layers were all able to perfectly learn the optimal transform to apply next for any given state.

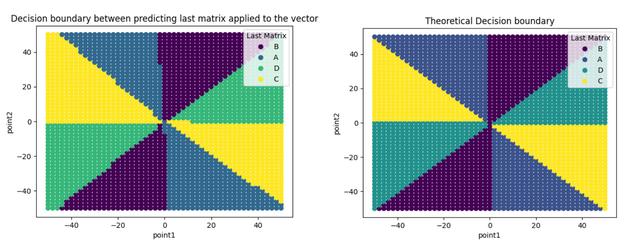

Fig. 1: Triangular regions match intuition

We then tested a deep MCTS implementation, which produced 95% accuracy with positive a, b, with no supervision.

Heisenberg Group

The Heisenberg Group is all 3x3 matrices of the form

$$\begin{bmatrix} 1 & a & c \\ 0 & 1 & b \\ 0 & 0 & 1 \end{bmatrix} \; a, b, c, \in \mathbb{Z}$$

The Heisenberg group is generated by the following four matrices:

$$A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}, \; B = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & -1 \\ 0 & 0 & 1 \end{bmatrix}, \; C = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \; D = \begin{bmatrix} 1 & -1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Problem: Given an element in the Heisenberg group, find the shortest string of A, B, C, and D which generates that element. We know that a path always exists to a given element in the Heisenberg Group, but it is not obvious what the fastest path is.

We generated data by using breadth-first search to explore all the possible steps for a matrix while trying to get that matrix back to the origin. Here is a visualization of the optimal first steps for when $a = 0$ (for when the entry in the upper middle is 0):

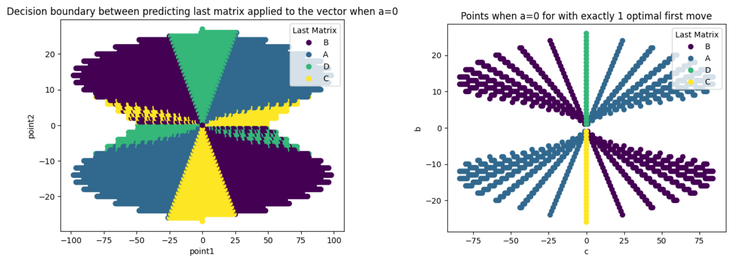

Fig. 2: Slices when a=0

85% of the points within 26 matrices of the origin have more than 1 optimal first step to take to get back to the origin.

Since predicting the next optimal move for any given state is nontrivial, generating training data is expensive, so we decided to apply reinforcement learning paradigms, rather than supervised learning algorithms.

We were able to achieve 97% accuracy with a tabular Q-learning algorithms on states up to 26 values from identity. We also tried applying deep Q-learning and deep MCTS, but neither performed very well.

We built a neural net atop the tabular Q-learning and it was able to achieve 70% accuracy for predicting the next matrix to apply. Here is a visual of what the neural network was learning to apply for the above slice where $a = 0$:

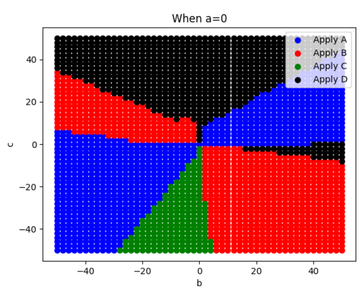

Fig. 3: Neural Network built atop Q-learning predictions

$SL_3(\mathbb{Z})$

$SL_3(\mathbb{Z})$ is the group of all 3x3 matrices with integer values and determinant 1. An open question from [1] about this group is whether it can be generated by the following matrices:

$$\begin{bmatrix} 1 & 1 & 2 \\ 0 & 1 & 1 \\ 0 & -3 & -2 \end{bmatrix} \begin{bmatrix} -2 & 0 & -1 \\ -5 & 1 & -1 \\ 3 & 0 & 1 \end{bmatrix}$$

We found that tabular Q-learning and neural networks both struggle to find solutions—it is unclear if this is because these matrices do not generate the group, or the architectures are not optimal for the task.

Summary and Future Work

We found that tabular Q-learning seems to perform well on "identity-finding" tasks—much better than deep Q-learning—probably because it is designed for discrete state and action spaces. However, its hypothesis space is finite, so its prediction space is always limited.

Deep MCTS remains a promising option, but our implementations need to be refined further. At the moment, training does not extrapolate well to predictions in environments with more possible actions, and is also computationally expensive.

In the future, we will continue investigating Q-learning's behavior on non-thin and thin subgroups in $SL_3\mathbb{Z}$. We will also continue investigating how the performance of Q-learning on $SL_3\mathbb{Z}$ relates to how much of the group is generated by our two matrices.

Code

Code we developed for this problem is publicly available online here:
https://github.com/dgconway/MXM-AI-Fall-2023

Acknowledgements

We would like to thank Karan Srivastava for his mentorship throughout this process. We would also like to thank Professor Jordan Ellenberg for his guidance and inspiration on this project.

References

[1] A. Kontorovich et al. "What is a Thin Group?" In: *American Mathematical Society* 66 (June 2019), pp. 905–910.