Machine Learning for Pure Math

Sophia Cohen, Noah Jillson, Pramana Saldin, Jimmy Vineyard, Alex Yun; Advised by Karan Srivastava, Prof. Jordan Ellenberg

University of Wisconsin - Madison

Introduction

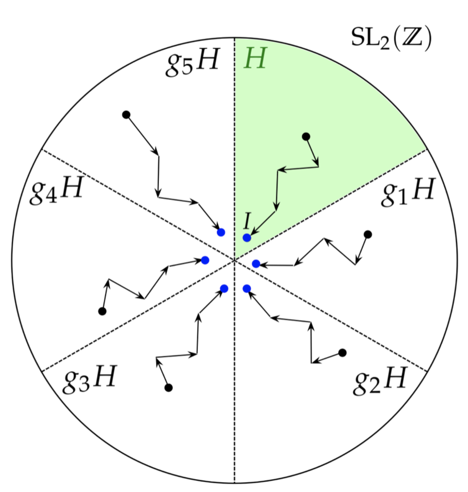
Computing the index of subgroups is a generally nontrivial task. This project aims to do so using various machine learning methods. Specifically, we focused on $\mathrm{SL}_2(\mathbb{Z})$, utilizing previous work addressing the same problem with the Heisenberg group.

We formulated the problem as follows: given a matrix in a subgroup, we continually apply generators to the matrix until we reach identity in as few steps as possible. We train a machine learning model to predict the optimal next generator to apply for any given element. We hoped this model would learn some inherent relationship between group elements

By analyzing which elements these models travel through when walking back from points in the full $\mathrm{SL}_2(\mathbb{Z})$ group, we hoped to identify core coset representatives in the subgroup that could assist in developing mathematical proofs about its index.

Mathematical Setup

Given a subgroup H of a group G, G can be partitioned into equally sized subsets g_iH that consist of all products obtained by multiplying a fixed element $g_i \in G$ with each element of the subgroup H. These subsets are called cosets and the number of cosets that G is partitioned into by a subgroup H is called the index of the group, which can be finite or infinite. We studied the subgroups A, B below.



$$A = \left\langle \begin{bmatrix} 1 & 2 \\ 0 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 0 \\ 2 & 1 \end{bmatrix} \right\rangle$$

$$B = \left\langle \begin{bmatrix} 1 & 3 \\ 0 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 0 \\ 3 & 1 \end{bmatrix} \right\rangle$$

Given a matrix $M \in SL_2(\mathbb{Z})$, we want to a) find an optimal decision rule $F(M)$ that walks back to the representative with the smallest ∞-norm in the corresponding coset (the "core") and b) detect the number of distinct representatives to find the index of the group. An equivalent problem is to study the action of the group on $\mathbb{Z}^2$, where we aim to get a decision rule that walks back to the vectors in the set $(\pm 1, 0)^T, (0, \pm 1)^T, (\pm 1, \pm 1)^T$.

We know several facts *a priori* about the group structure for the following subgroups of $\mathrm{SL}_2(\mathbb{Z})$. In particular, we know that A has index 12, and B has infinite index. We also know that they are free groups, meaning the optimal path to an element of the group is unique.

Machine Learning Methods

Our most successful approach used a four layer neural network that learned to output the best move towards the origin, given a vector or matrix. We trained two different models that achieved 96% accuracy for the two subgroups.

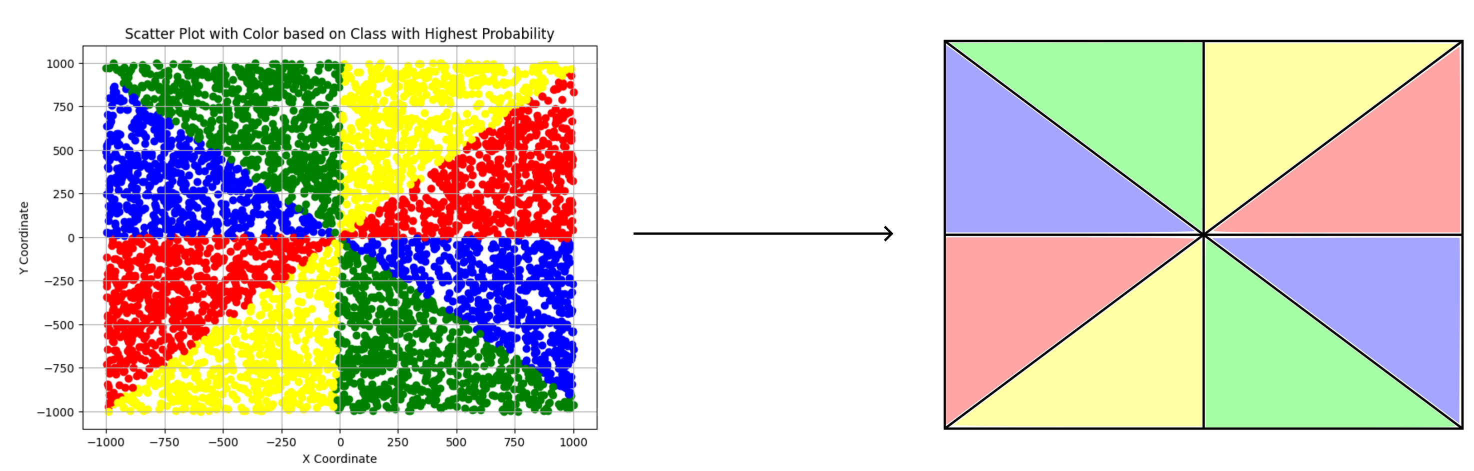
We generated training data by doing a random walk out from an origin vector $(1,0)^T$ using the two generators and their inverses. The last generator used was saved as the "best move" label for each state.

Machine Learning Methods (cont.)

Because this supervised learning approach requires some inherent knowledge about the groups, we also tried applying deep reinforcement learning (DRL) *a priori* to this problem. In particular, we applied a deep Q-learning network [3, 1] and a deep Monte Carlo Tree Search [4] to the task. However, neither of these performed nearly as well as the supervised approach.

Results

The goal of this project was to use machine learning to assist us with creating proofs in pure math. We noticed that the decision boundaries that the machine was learning were approximately linear, so we created a set of ideal linear decision boundaries that we thought the machine was learning.



Machine Learning Algorithm — Human Interpretation

Fig. 2: We inferred that the machine was learning a linear decision boundary.

Let $F\colon \mathbb{Z}^2 \to \{T, T^{-1}, U, U^{-1}\}$ be the function that chooses what matrix to right-multiply to $\mathbf{x} = (x_1, x_2)$ to get to the core as quickly as possible. The machine learning boundaries indicate that F should be defined as

$$F(\mathbf{x}) = \begin{cases} U & |x_1| < |x_2|, x_1x_2 < 0, \\ U^{-1} & |x_1| < |x_2|, x_1x_2 \geq 0, \\ T & |x_1| \geq |x_2|, x_1x_2 < 0, \\ T^{-1} & |x_1| \geq |x_2|, x_1x_2 \geq 0. \end{cases}$$

Theorem 1. *Let* $\mathbf{x} = (x_1, x_2)$ *where* $|x_1| \neq |x_2|$ *and* $x_1 \neq 0$ *and* $x_2 \neq 0$. *Then*

$$\|F(\mathbf{x})\mathbf{x}\| < \|\mathbf{x}\|. \tag{1}$$

Otherwise,

$$\|F(\mathbf{x})\mathbf{x}\| = \|\mathbf{x}\| \leq 1. \tag{2}$$

Proof (sketch). We first prove (1). The conditions imply that one of x_1, x_2 has absolute value greater than or equal to 2. Applying the matrix $F(\mathbf{x})$ reduces the entry with higher absolute value to something below the absolute value of the other entry.
Since $\mathbf{x}$ is the first column of some matrix in $\mathrm{SL}_2(\mathbb{Z})$, this significantly restricts the possibilities where $|x_1| = |x_2|$ or $x_0 = 0$ or $x_1 = 0$ (i.e. where (2) applies). It turns out this only happens when $\|\mathbf{x}\| = 1$. So applying $F(\mathbf{x})$ always reduces the norm of $\mathbf{x}$ unless it has norm 1. ☐

This theorem implies that the decision rule F *always* returns to the core. Hence, the machine learned an approximation of this algorithm that works.

The same machine learning architectures successfully walked back to the core of A, and also the following core elements of B:

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix}, \begin{bmatrix} 0 & 1 \\ -1 & -1 \end{bmatrix}, \begin{bmatrix} 0 & 1 \\ -1 & 1 \end{bmatrix}, \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}, \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}, \begin{bmatrix} 1 & -1 \\ 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} -1 & 1 \\ 0 & -1 \end{bmatrix}, \begin{bmatrix} -1 & 0 \\ 1 & -1 \end{bmatrix}, \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}, \begin{bmatrix} 1 & 1 \\ -1 & 0 \end{bmatrix}, \begin{bmatrix} -1 & 1 \\ -1 & 0 \end{bmatrix}, \begin{bmatrix} 0 & -1 \\ 1 & 1 \end{bmatrix}, \begin{bmatrix} 1 & -1 \\ 1 & 0 \end{bmatrix}, \begin{bmatrix} -1 & -1 \\ 1 & 0 \end{bmatrix}, \begin{bmatrix} -1 & 0 \\ -1 & -1 \end{bmatrix}$$

Heisenberg Group Conjectures

The Heisenberg group is all 3x3 matrices of the form

$$[a,b,c]_H := \begin{bmatrix} 1 & a & c \\ 0 & 1 & b \\ 0 & 0 & 1 \end{bmatrix} \qquad a, b, c, \in \mathbb{Z}$$

Based on what the machine learning algorithms predicted last semester and this semester, we came up with the following conjectures:

Conjecture 1. *There is no finite set of linear forms* $L_i : \mathbb{R}^3 \to \mathbb{R}$ *such that the optimal generator application to get to the identity for* $[a,b,c]_H$ *is determined by the inequalities between the* $L_i(a,b,c)$.

Conjecture 2. *We define a decision rule* L *to be homogeneous if* $L(a,b,c) = L(ka,kb,kc)$ *for* $k \in \mathbb{Z}_{>0}$. *There is no homogeneous decision rule which gives the optimal move for all elements of the Heisenberg group.*

Conjecture 3. *Let* C *be the Cayley graph of* H. *Let* C' *be the subgraph of* C *generated by only including edges between adjacent vertices with the same* ℓ_∞ *norm. Let* $X \subset C'$ *be a connected component of* C'. *Then, there exists some* $x \in X$ *such that the corresponding* $x \in C$ *is adjacent to a point with lower* ℓ_∞ *norm.*

Some of these conjectures do not seem to be the case. For example, experimental data using a breadth first search algorithm has failed to find a counterexample that would prove conjecture 2.

Future Directions

We would like to extend our models to $\mathrm{SL}_3(\mathbb{Z})$. For instance, the following conjecture is unsolved:

Conjecture 4 ([2]). *Does the following subgroup have infinite or finite index?*

$$\Gamma := \left\langle \begin{bmatrix} 1 & 1 & 2 \\ 0 & 1 & 1 \\ 0 & -3 & -2 \end{bmatrix}, \begin{bmatrix} -2 & 0 & -1 \\ -5 & 1 & -1 \\ 3 & 0 & 1 \end{bmatrix} \right\rangle \leq \mathrm{SL}_3(\mathbb{Z})$$

A decision rule that always brings all matrices to some ball around the origin shows that this subgroup has finite index.

We would also like to try applying some other machine learning architectures to this problem, including using transformers.

Code

The code we developed for this project is publicly available online at:
`https://github.com/MXM-MachineLearning/MXM_AI_Ellenberg`

References

[1] Hado van Hasselt, Arthur Guez, and David Silver. *Deep Reinforcement Learning with Double Q-learning.* 2015. arXiv: `1509.06461 [cs.LG]`.

[2] Alex Kontorovich et al. "WHAT IS...a Thin Group?" In: *Notices of the American Mathematical Society* 66 (June 2019), p. 1. DOI: `10.1090/noti1900`.

[3] Volodymyr Mnih et al. *Playing Atari with Deep Reinforcement Learning.* 2013. arXiv: `1312.5602 [cs.LG]`.

[4] David Silver et al. "Mastering the game of Go without human knowledge". In: *Nature* 550.7676 (2017), pp. 354–359.