Solving Algorithmic Problems in Algebraic Structures via Machine Learning

Jonathan Gryak
University of Michigan, Ann Arbor

ICMS 2018
University of Notre Dame
July 26, 2018

How Machine Learning Can Help

1. Solving Decision Problems
2. Exploring Algebraic (Sub)structure
3. Choosing the Best Algorithm for a Particular Class of Structure
4. Heuristics/Metrics for Search Problems

Methods of Studying Groups

A number of methods have been developed:

- Linear Representations
- Computational
- Combinatorial
- Geometric
- Machine Learning [5]

We wish to apply machine learning techniques to solving algorithmic problems in non-free infinite groups.

Related Work

- In [5], Haralick, et al. suggested a machine learning approach to solving algorithmic problems in free groups.
- Used supervised learning and clustering to investigate the Whitehead minimization problem
- Provided a general framework for the application of machine learning techniques to group-theoretic problems.

Tasks Required for Supervised Machine Learning

- Data Generation
- Feature Extraction
- Model Selection
- Evaluation and Analysis

Data Sets

In supervised learning, the goal is to train a model that can predict classes or values on new, unseen members of the data domain. Therefore, we need at least two distinct sets - training and verification. We suggest three:

- *Training Set* - The set S_i is used to train the initial classifier or regression function.
- *Optimization Set* - The set S_o can used to optimize the parameters of a decision rule, or be used for feature selection.
- *Verification Set* - The set S_v is used to evaluate the performance of the (optimized) decision rule.

If generating data is expensive or data is sparse, cross-validation strategies such as *k-fold cross validation* can be employed.

Feature Extraction via Counting Subgraphs and Normal Forms

- Given a free group $F(X)$ and a finite set of words $U = \{u_1, \ldots, u_k \mid u_j \in F(X)\}$, for any word $w \in F(X)$ we can form a *counting subgraph* $\Gamma(w) = (V, E)$, with $V = X \cup X^{-1}$
- For any $x, y \in V$ and $u_j \in U$, we form a directed edge $(x, y) \in E$, labeled by xu_jy and assigned the weight $C(w, xu_jy)$, which is equal to the number of times the reduced subword xu_jy occurs in w
- These are directed generalization of Whitehead graphs
- Features can be extracted by combining different *counting functions* $C(w, t)$ into feature vectors
- If a finitely presented group possesses an efficiently computable *normal form*, we can readily convert these forms to feature vectors

Feature Vectors

Let G be a finitely generated group with generating set X and possessing a normal form, and let $Y = X \cup X^{-1}$.

- n_0 (*Normal Form*) - If w is a word in normal form, then w is of the form
$$y_1^{e_1} \cdots y_N^{e_N}$$
with $y_i \in Y$ and $e_i \in \mathbb{Z}$. The feature vector n_0 is then
$$n_0 = \langle e_1, \ldots, e_N \rangle.$$
- n_1 (*Weighted Normal Form*) - The feature vector n_1 is the same as n_0 above, except it is weighted by the word length $|w|$:
$$n_1 = \frac{1}{|w|} \langle e_1, \ldots, e_N \rangle.$$

Feature Vectors (cont.)

The features below were introduced in [5]. They apply to finitely presented groups in general and do not require a normal form:

- f_0 (*Generator Count*) - Given a fixed order on X, let $x_i \in X$ be the ith generator. We can then define the counting function $C(w, x_i) = |\{w_j \mid w_j = x_i \vee x_i^{-1}\}|$.The feature vector f_0 is then

$$f_0 = \langle C(w, x_1), \ldots, C(w, x_N) \rangle.$$

- f_1 (*Weighted Generator Count*) - The feature vector f_1 is the same as f_0 above, except it is weighted by the word length $|w|$:

$$f_1 = \frac{1}{|w|} \langle C(w, x_1), \ldots, C(w, x_N) \rangle.$$

Feature Vectors (cont.)

- f_2 through f_7 (*Counting Subgraphs*) - Let $U_l = \{u_j \in F(X) \mid |u_j| = l\}$, and consider the counting functions $C(w, xu_{lj}y)$, with $u_{lj} \in U_l$ and $x, y \in X$ such that $xu_{lj}y$ is a geodesic word. For each subword length l there is a weighted and non-weighted variant:

$$
\begin{aligned}
f_2 &= \langle C(w, xu_{1j}y) \mid x, y \in Y; u_j \in U_1 \rangle \\
f_3 &= \frac{1}{|w|}\langle C(w, xu_{1j}y) \mid x, y \in Y; u_j \in U_1 \rangle \\
f_4 &= \langle C(w, xu_{2j}y) \mid x, y \in Y; u_j \in U_2 \rangle \\
f_5 &= \frac{1}{|w|}\langle C(w, xu_{2j}y) \mid x, y \in Y; u_j \in U_2 \rangle \\
f_6 &= \langle C(w, xu_{3j}y) \mid x, y \in Y; u_j \in U_3 \rangle \\
f_7 &= \frac{1}{|w|}\langle C(w, xu_{3j}y) \mid x, y \in Y; u_j \in U_3 \rangle
\end{aligned}
$$

Model Selection

- *Many* models to choose from (e.g., SVM, Naïve Bayes, CNN)
- Explore models that are discrete and whose results are interpretable:
  - Decision Tree - Uses a tree structure to partition the feature space
  - Random Forest - Uses an ensemble of trees with subsampling of the feature vector
  - N-Tuple Neural Network (NTNN) - Aggregate observed subsamples of the feature vector

N-Tuple Neural Networks (NTNN)

NTNNs can be interpreted through the framework of relational algebra:

- Transform each sample s into a feature vector x of length N
- Let $J_1, \ldots, J_M$ be index sets or *patterns* of uniform length P, that is, subsets of the set $\{0, \ldots, N-1\}$
- Maintain tables T_{mc} - one for each pattern J_m and class $c \in C$
- Given a feature vector x, the projection $k = \pi_{mc}(x)$ is stored in T_{mc}
- For each observed value k, a count c_k of the number of times k was observed is stored in T_{mc}
- During classification, assign s to class c' if $\sum_{m \in M} T_{mc'}(s) > \sum_{m \in M} T_{mc}(s)$ for all classes $c \neq c'$.
- Choice of patterns can be optimized through a greedy algorithm

NTTN - Example

Consider an NTNN with the parameters $N = 5$, $M = 2$, and $P = 3$. The table below represents the NTNN table entries for class 0 after training on the first 3 samples:

s	J_0	π_{00}	T_{00}
$\langle -4, -1, 5, 2, 3 \rangle$ $\langle -4, -7, 5, 2, 3 \rangle$ $\langle -2, -1, 6, 3, 1 \rangle$	$(0, 2, 4)$	$(-4, 5, 3)$ $(-4, 5, 3)$ $(-2, 6, 1)$	$(-4, 5, 3) \mapsto 2$ $(-2, 6, 1) \mapsto 1$
s	J_1	π_{10}	T_{10}
$\langle -4, -1, 5, 2, 3 \rangle$ $\langle -4, -7, 5, 2, 3 \rangle$ $\langle -2, -1, 6, 3, 1 \rangle$	$(1, 2, 3)$	$(-1, 5, 2)$ $(-7, 5, 2)$ $(-1, 6, 3)$	$(-1, 5, 2) \mapsto 1$ $(-7, 5, 2) \mapsto 1$ $(-1, 6, 3) \mapsto 1$

Evaluation and Analysis

If the classes are balanced, then accuracy is the preferred measure of classification performance. The accuracy $\mathcal{A}$ of the model $\mathcal{M}(f)$, trained with respect to the feature vector f, over the test set S_v, is calculated as:

$$\mathcal{A}(\mathcal{M}(f), S_v) = \frac{|\text{True Positives}(S_v)| + |\text{True Negatives}(S_v)|}{|S_v|}$$

Solving the CDP via Machine Learning

Solving the Conjugacy Decision Problem via Machine Learning[3]

Paper

Jonathan Gryak, Robert M. Haralick, and Delaram Kahrobaei. Solving the Conjugacy Decision Problem via Machine Learning. *Experimental Mathematics, 1–13, 2018.*

Conjugacy Decision Problem

Conjugacy Decision Problem

Given a group G and elements $u, v \in G$, the *conjugacy decision problem* asks if $\exists z \in G$ such that $u = zvz^{-1}$

Data Generation

Each dataset consists of 20,000 pairs of words in normal form, with two 10,000 pair halves that are generated via the following procedures:

1. *Random Non-Conjugate Word Pairs in Normal Form* - For each $n \in [5, 1004]$ we generate two words u, v representing elements in G, with $|u| = |v| = n$. To verify that u is not conjugate to v, the method of Kapovich et al. [6] is used.
2. *Random Conjugate Word Pairs in Normal Form* - For $n \in [5, 1004]$ we generate a pair of words v, z representing element in G with $|v| = |z| = n$. Each word v, z is generated uniformly and randomly as above. After v and z are generated, the word $u = v^z$ is formed, and the tuple (u, v) is added to the dataset.

This process is repeated 10 times for each n.

Additional Datasets

To better evaluate the performance of our classifiers, we generated additional data sets with varying ranges of lengths:

Collection	**Conjugate Pair** $(u, v = u^t)$	**Non-Conjugate Pair** (u, v)
D_0	$\lvert u\rvert = \lvert t\rvert = l; \quad l \in [5, 1004]$	$\lvert u\rvert = \lvert v\rvert = m; \quad m \in [5, 1004]$
D_1	$\lvert u\rvert = \lvert t\rvert = l; \quad l \in [5, 1004]$	$\lvert u\rvert = m, \lvert v\rvert = n; \quad m, n \in [5, 1004]$
D_2	$\lvert u\rvert = l, \lvert t\rvert = p; \quad l, p \in [5, 1004]$	$\lvert u\rvert = m, \lvert v\rvert = n; \quad m, n \in [5, 1004]$
D_3	$\lvert u\rvert = l, \lvert t\rvert = p; \quad l, p \in [5, 1004]$	$\lvert u\rvert = m, \lvert v\rvert = n; \quad m \in [5, 1004]$ $n \in [\min_{D_2}, \max_{D_2}]$

where $\min_{D_2}$ $(\max_{D_2})$ correspond to the minimum (maximum) word length in a conjugate pair for the data set D_2

Feature Vectors

Given a group G and words $u, v \in G$, we concatenate the unit feature vectors n_0 and n_1 to create two derived feature vectors for the conjugacy decision problem:

$$c_0 = \langle n_0(u) \parallel n_0(v) \rangle$$
$$c_1 = \langle n_1(u) \parallel n_1(v) \rangle$$

The default feature vector for tree-based classifiers is c_1 (weighted normal forms), while for NTNNs it is c_0 (unweighted normal forms).

Model Selection

We tested all three classification models with the following parameters:

- Decision Tree and Random Forests:
  - Both Gini impurity and information gain were evaluated.
  - Having no depth limit or pre-pruning to a depth of $log_2 S_i - 1$ were both tested.
- NTNN:
  - The number of patterns M was tested over the range $\{10, 20, 30, 50, 100\}$.
  - The initial size P of the patterns was set to 3, with sizes in the range $[3, 5]$ tested where applicable.

BS(1,2)

The Baumslag-Solitar group BS(1,2) is given by the presentation below:

$$BS(1,2) = \langle a, b \mid bab^{-1}a^{-2} \rangle.$$

The group has the normal form

$$n_0 = b^{-e_1} a^{e_2} b^{e_3},$$

with $e_1, e_3 \geq 0$ and if $e_1, e_3 > 0$ then e_2 is not divisible by 2.

Non-Virtually Nilpotent Polycyclic Groups

These non-virtually nilpotent polycyclic groups can be constructed by using the `MaximalOrderByUnitsPcpGroup` function of the GAP Polycyclic package [2]:

- $\mathcal{O} \rtimes U_{14}$ - Given the polynomial $f = x^9 - 7x^3 - 1$, `MaximalOrderByUnitsPcpGroup` returns a group with a Hirsch length of 14.
- $\mathcal{O} \rtimes U_{16}$ - Given the polynomial $f = x^{11} - x^3 - 1$, `MaximalOrderByUnitsPcpGroup` returns a group with a Hirsch length of 16.
- $\mathcal{O} \rtimes U_{34}$ - Given the polynomial $f = x^{23} - x^3 - 1$, `MaximalOrderByUnitsPcpGroup` returns a group with a Hirsch length of 34.

GMBS(2,3)

The generalized metabelian Baumslag-Solitar[4] group GMBS(2,3) given by the following presentation:

$$\mathrm{GMBS}(2,3) = \langle q_1, q_2, b \mid b^{q_1} = b^2, b^{q_2} = b^3, [q_1, q_2] = 1 \rangle.$$

Elements in GMBS(2,3) can be uniquely written in the following normal form:

$$n_0 = q_1^{-e_1} q_2^{-e_2} b^{e_3} q_1^{e_4} q_2^{e_5},$$

with $e_1, e_2, e_4, e_5 \geq 0$, $2 \nmid e_3$ if $e_1, e_4 > 0$, and $3 \nmid e_3$ if $e_2, e_5 > 0$.

$\mathrm{SL}(2,\mathbb{Z})$

The set of 2×2 integral matrices with determinant 1 forms the group $\mathrm{SL}(2,\mathbb{Z})$ under matrix multiplication.

$\mathrm{SL}(2,\mathbb{Z})$ was implemented in `GAP` with a dual representation: For each element $x \in \mathrm{SL}(2,\mathbb{Z})$ we have the following pair $x = (m, w)$ of the form

$$m = \begin{bmatrix} a & b \\ c & d \end{bmatrix}, w = w_1 \cdots w_n, w_i \in \{S^{\pm 1}, R^{\pm 1}\},$$

with $a, b, c, d \in \mathbb{Z}$ such that $ad - bc = 1$, and S and R corresponding to the matrices below that generate $\mathrm{SL}(2,\mathbb{Z})$:

$$S = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}, R = \begin{bmatrix} 0 & -1 \\ 1 & 1 \end{bmatrix}.$$

Performance on Different Datasets

Despite the changes in word lengths within each data collection, classification accuracy was maintained:

	Data Collection			
Group	$\mathbf{D_0}$	$\mathbf{D_1}$	$\mathbf{D_2}$	$\mathbf{D_3}$
BS(1,2)	93.64% (F_e)	93.20% (F_g)	95.30% (F_e)	98.86% (F_e)
$\mathcal{O} \rtimes U_{14}$	98.77% (N_l)	98.67% (F_{ed})	98.38% (F_{ed})	99.75% (N_l)
$\mathcal{O} \rtimes U_{16}$	98.46% (N_s)	97.24% (F_{ed})	96.65% (F_{ed})	99.11% (F_g)
$\mathcal{O} \rtimes U_{34}$	99.50% (N_l)	98.72% (F_{ed})	98.28% (F_{ed})	99.29% (N_l)
GMBS(2,3)	96.49% (F_e)	95.22% (F_e)	96.45% (N_s)	99.13% (F_{gd})
$\mathrm{SL}(2,\mathbb{Z})$	99.81% (N_l)	99.91% (F_g)	93.89% (F_e)	97.38% (F_g)

Further Applications

Thus we have shown the following:

- A general method for applying machine learning to problems in non-free groups
- The conjugacy decision problem can be solved using machine learning with high accuracy

Further applications:

2. Exploring Algebraic (Sub)structure
3. Choosing the Best Algorithm
4. Heuristics/Metrics for Search Problems

Exploring Algebraic Structure

- Use discrete, interpretable supervised learning models (e.g., NTNN, Decision Trees) for classification
- Use unsupervised learning models for clustering (e.g., K-means clustering)
- Use higher order features to explore superstructure

Choosing the Best Algorithm

- Use classifiers to determine whether an object is a member of a particular class with better algorithms (e.g., hyperbolic group, automatic group)

Heuristics/Metrics for Search Problems

- Use metric learning (e.g., ITML [1]) to determine lengths in an appropriate space
- For search problems, use regression models (e.g., NTRN) to produce estimate that can act as an initial seed for heuristic-based search (e.g., local conjugacy search over the Cayley graph)

Questions?

Your questions and comments, please.

References

[1] Jason V Davis, Brian Kulis, Prateek Jain, Suvrit Sra, and Inderjit S Dhillon. Information-theoretic metric learning. In *Proceedings of the 24th international conference on Machine learning*, pages 209–216. ACM, 2007.

[2] Bettina Eick, Werner Nickel, and Max Horn. Polycyclic, computation with polycyclic groups, Version 2.11. http://www.icm.tu-bs.de/ag_algebra/software/polycyclic/, Mar 2013. Refereed GAP package.

[3] Jonathan Gryak, Robert M. Haralick, and Delaram Kahrobaei. Solving the conjugacy decision problem via machine learning. *Experimental Mathematics*, pages 1–13, 2018.

References (cont.)

[4] Jonathan Gryak, Delaram Kahrobaei, and Conchita Martinez-Perez. On the conjugacy problem in certain metabelian groups. 2018. Glasgow Mathematical Journal, Cambridge University Press.

[5] Robert Haralick, Alex D. Miasnikov, and Alexei G. Myasnikov. Pattern recognition approaches to solving combinatorial problems in free groups. *Computational and Experimental Group Theory: AMS-ASL Joint Special Session, Interactions Between Logic, Group Theory, and Computer Science, January 15-16, 2003, Baltimore, Maryland*, 349:197–213, 2004.

[6] Ilya Kapovich, Alexei G. Myasnikov, Paul Schupp, and Vladimir Shpilrain. Generic-case complexity, decision problems in group theory, and random walks. *Journal of Algebra*, 264(2):665–694, 2003.