Pacific Northwest
NATIONAL LABORATORY

PNNL- 36691

Machine Learning meets Algebraic Combinatorics

A Suite of Benchmark Datasets to Accelerate AI for Mathematics Research

September 2024

Herman Chau
Helen Jenne
Davis Brown
Sara Billey
Mark Raugas
Henry Kvinge

U.S. DEPARTMENT OF ENERGY

Prepared for the U.S. Department of Energy
under Contract DE-AC05-76RL01830

DISCLAIMER

PACIFIC NORTHWEST NATIONAL LABORATORY
operated by
BATTELLE
for the
UNITED STATES DEPARTMENT OF ENERGY
under Contract DE-AC05-76RL01830

Printed in the United States of America

Available to DOE and DOE contractors from
the Office of Scientific and Technical Information,
P.O. Box 62, Oak Ridge, TN 37831-0062
www.osti.gov
ph: (865) 576-8401
fox: (865) 576-5728
email: reports@osti.gov

Available to the public from the National Technical Information Service
5301 Shawnee Rd., Alexandria, VA 22312
ph: (800) 553-NTIS (6847)
or (703) 605-6000
email: info@ntis.gov
Online ordering: http://www.ntis.gov

PNNL- 36691

Machine Learning meets Algebraic Combinatorics

A Suite of Benchmark Datasets to Accelerate AI for Mathematics Research

September 2024

Herman Chau
Helen Jenne
Davis Brown
Sara Billey
Mark Raugas
Henry Kvinge

Prepared for
the U.S. Department of Energy
under Contract DE-AC05-76RL01830

Pacific Northwest National Laboratory
Richland, Washington 99354

Abstract

The use of benchmark datasets has become an important engine of progress in machine learning (ML) over the past 15 years. Recently there has been growing interest in utilizing machine learning to drive advances in research-level mathematics. However, off-the-shelf solutions often fail to deliver the types of insights required by mathematicians. This suggests the need for new ML methods specifically designed with mathematics in mind. The question then is: what benchmarks should the community use to evaluate these? On the one hand, toy problems such as learning the multiplicative structure of small finite groups have become popular in the mechanistic interpretability community whose perspective on explainability aligns well with the needs of mathematicians. While toy datasets are a useful benchmark for initial work, they lack the scale, complexity, and sophistication of many of the principal objects of study in modern mathematics. To address this, we introduce a new collection of benchmark datasets, Algebraic Combinatorics Benchmarks (ACBench), representing either classic or open problems in algebraic combinatorics, a subfield of mathematics that studies discrete structures arising from abstract algebra. After describing the datasets, we discuss the challenges involved in constructing “good” mathematics benchmarks, describe baseline model performance, and discuss some of the insights these datasets can provide that may be of interest even to those who are not interested in mathematics research itself.

Acknowledgments

This research was supported by the Mathematics for Artificial Reasoning in Science (MARS) initiative at Pacific Northwest National Laboratory, under the Laboratory Directed Research and Development (LDRD) Program at Pacific Northwest National Laboratory (PNNL). PNNL is a multi-program national laboratory operated for the U.S. Department of Energy (DOE) by Battelle Memorial Institute under Contract No. DE-AC05-76RL01830.

Contents

Figures

1.0 Introduction

Modern approaches to machine learning (ML) have been shown to be capable of extracting sophisticated patterns from large and complex datasets. As this capability has grown, there is increasing evidence that ML can be used as a tool to advance scientific discovery. Notable successes include AlphaFold [20] for predicting protein folding and machine learning for event generation and simulation-based inference in high-energy physics [10]. Beyond the natural sciences, a growing community of researchers are also looking at ML as a tool to aid the research mathematician. Some of this research explores the use of LLMs and related models to aid in proof writing and higher mathematical reasoning [34,40]x, but there is also a need for models to help analyze (what we call) ‘raw' mathematical data. This data, which often takes the form of (very) long lists of examples, is used by the mathematician to develop intuition and formulate conjectures. Though the popular perception is that research mathematics takes place at a level of abstraction beyond individual instances, the manual examination of examples (data) constitutes a fundamental part of the mathematician’s workflow for many problems. For example, when trying to better understand the coefficients of a particular family of polynomials (e.g., Kazhdan-Lusztig polynomials in Section 2.0), a mathematician may look through countless examples to either generate conjectures or build evidence for existing conjectures that they have.

Existing applications of machine learning to raw mathematics data tend to fall into one of two types. The first are toy problems that are simple enough that we can hope to explicitly describe how the model is solving the problem. Example tasks include: modular addition [40, 29], multiplication of small symmetric groups [28], and other arithmetic tasks [27]. These are used by AI researchers in the interpretability community (particularly mechanistic interpretability) as small, self-contained, and tractable problems where at least one solution is very well understood. By design these do not represent (or even aim at) open problems in mathematics. The other cluster of works focus on solving very specific problems in research-level mathematics [13,39]. These works tend to come from the mathematics community and often come with a high knowledge barrier, requiring the reader to already be relatively familiar with the underlying domain (an exception is [45]). For these, machine learning is often just one of several tools used when solving the problem of interest and is not the primary contribution of the paper.

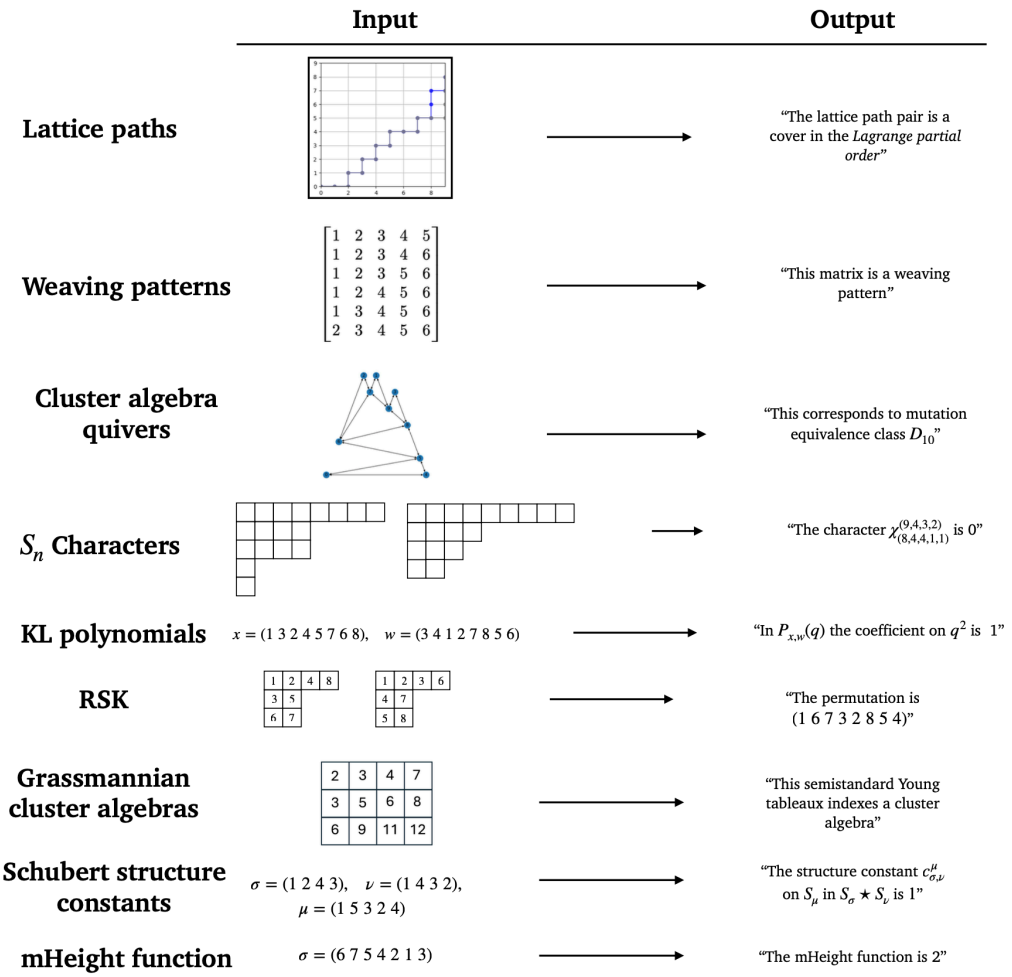

Figure 1: A visualization of the tasks included in ACBench.

Unfortunately, neither of these types of works satisfy the needs of a benchmark. While toy tasks like modular arithmetic are ideal for mechanistic interpretability research, they do not represent the kinds of problems that mathematicians work on. On the other hand, papers coming from the mathematics community each use different datasets and require extensive background knowledge to understand. To fill the need for accessible benchmarks that represent modern mathematics research but are structured for machine learning research we present *Algebraic Combinatorics Benchmarks (ACBench)*[1] a collection of 9 datasets designed for the development of machine learning tools to advance research mathematics. Our benchmark includes both open problems and classic problems whose solution is a major result in the field. Algebraic combinatorics is an area of mathematics that studies discrete structures arising from abstract algebra (including algebraic geometry and representation theory). We chose to focus on this domain because: (i) it requires less background theory to understand, making it generally more accessible than, for instance, algebraic topology, differential geometry, or analysis, while still remaining absolutely fundamental to cutting-edge mathematics, (ii) there already exist specialized software libraries (e.g., Sage [36]) designed to efficiently compute many quantities

[1] Datasets and associated code can be found at https://github.com/pnnl/ML4AlgComb

of interest in algebraic combinatorics, and (iii) by nature of being discrete, the objects of interest in algebraic combinatorics tend to be more amendable to representation on a computer.

We believe that three communities will find this benchmark useful. The most obvious are researchers from either machine learning or mathematics that are interested in accessible problems that can be used when developing methods for mathematics research. The second is interpretability researchers, since finding a solution to an open problem through the interpretation of a performant neural network would be a major accomplishment for the field. Finally, 'science of deep learning' researchers may be interested in datasets representing complex tasks for which one can generate an arbitrary amount of data and make data of arbitrary input dimension and complexity, adding to existing resources in this direction (e.g., [38]).

In summary our contributions in this paper include

- The introduction of the Algebraic Combinatorics Benchmark which contains a range of machine learning tasks around both open problems and seminal results in the mathematical discipline of algebraic combinatorics.
- Descriptions of all datasets including relevant background, mathematical context, open problems around the dataset (if any) and the significance of existing solutions (if not).
- A discussion of the challenges involved in developing mathematics-based ML benchmarks and interesting aspects of training models on ACBench datasets such as dependence on size parameters and apparent difficulty of each task.

1.1 Background and Datatypes

The field of combinatorics studies a broad range of problems in mathematics centered around discrete objects (e.g, partial orders, graphs, permutations, partitions) [35,36]. Ideas and tools from combinatorics play an essential role in many other fields of mathematics and continue to have a strong impact on computer science and physics. Algebraic combinatorics is a subfield of combinatorics that applies combinatorial methods to problems arising from abstract algebra, particularly algebraic geometry and representation theory. In this section we review datatypes that play a central role in the field and that appear in ACBench.

Partitions: We use the word partition in this work to mean an integer partition. An integer partition of $n \in \mathbb{N}_{>0}$ is a sequence of positive integers $(n_1, n_2, \ldots, n_k)$ such that $n = n_1 + n_2 + \cdots + n_k$ and $n_1 \geq n_2 \geq \cdots \geq n_k$. We use the standard notation $\mu \vdash n$ to denote that μ is a partition of n. A partition $(n_1, \ldots, n\_k)$ is often visualized as a *Young diagram*, with n_1 left justified square cells in the first row, n_2 left justified square cells in the second row, etc. See Figure 2.

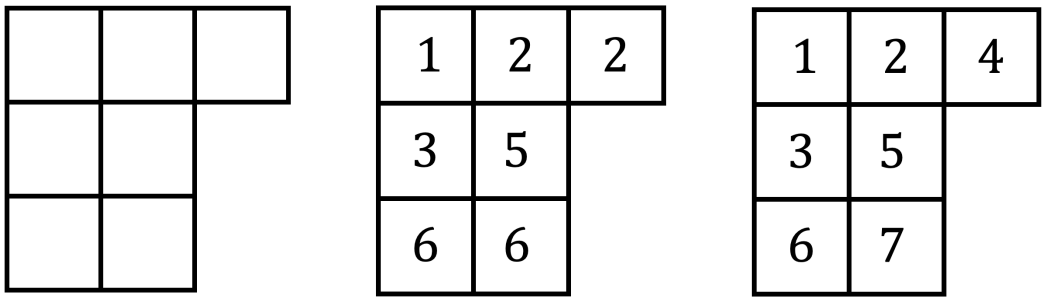

Figure 2: **(Left)** A Young diagram for the partition $(3,2,2)$. **(Center)** A standard Young tableaux for the partition $(3,2,2)$. **(Right)** A semistandard Young tableaux for the partition $(3,2,2)$.

Young tableaux: We noted above that a partition can be visualized as a Young diagram. Surprisingly, including extra decorations on the cells in a Young diagram can capture fundamental combinatorics in representation theory and other fields. A *Young tableau* corresponding to a Young diagram $\lambda \vdash n$ is a labeling of the cells of λ by an alphabet of symbols. In this work we will consider two types of Young tableau. A *standard Young tableau* corresponding to partition $\lambda \vdash n$ is a labeling of the cells of λ by $1, 2, \dots, n$ such that the integers strictly increase as one moves down a column or left to right across a row. See Figure 2 (center) for an example of a standard Young tableau for the partition $(3,2,2)$ of 7. The definition of a *semistandard Young tableau* is analogous except that the entries are only assumed to weakly increase as one moves from left to right along a row (see Figure 2 (right)).

Permutations: Permutations are familiar in machine learning from their central role in computer science as well as their relevance to symmetries in many neural networks [16,2,19] and as a symmetry in graphs [24] and set-based problems [40,26]. There are many ways to represent a permutation. In this paper we use *one-line notation*, which is best illustrated through an example. Suppose that ω is the permutation of the set of elements $\{1,2,3,4\}$ that swaps 1 and 2 and 3 and 4. Then in one-line notation we would write $\omega = 2\ 1\ 4\ 3$. 2 is in the first position since 1 is sent to 2, 1 is in the second position since 2 is sent to 1, 4 is in the third position since 3 is sent to 3, and 3 is in the fourth position since 4 is sent to 3.

Permutations can be written as sequences of transpositions of adjacent elements. For instance, the permutation $\sigma = 3\ 1\ 2$ can be formed by swapping $1\ 2\ 3 \to 1\ 3\ 2 \to 3\ 1\ 2$. If we denote a transposition of the ith and $(i+1)$st element as s_i and read from right to left (as is the convention) then σ can be written as $s_1 s_2$. A sequence of adjacent transpositions $s_{i_1} s_{i_2} \dots s_{i_k}$ corresponding to a permutation σ is called a *reduced word* if there is no other representations that uses fewer than k adjacent transpositions to represent σ. Finally, two reduced words are considered *commutation equivalent* if one can be obtained from another by swaps of the form $s_i s_j \mapsto s_j s_i$ where $|i-j| > 1$. A *descent* in a permutation $\sigma = a_1 a_2 \dots a_n$ is a pair (a_i, a_j) such that $i < j$ but $a_i > a_j$. The *descent set* of σ is simply the set of all descents. A related notion is that of a 3412 *pattern*. This is a quadruple (a_i, a_j, a_k, a_ℓ) such that $i < j < k < \ell$ but $a_k < a_\ell < a_i < a_j$. Descents and patterns have deep connections to algebra and geometry.

In the discussion above we implicitly thought of permutations of n as bijective functions from $\{1,2,\dots,n\} \to \{1,2,\dots,n\}$. Using this perspective, one can define the composition of two permutations. The symmetric group, denoted S_n, is defined as the group of permutations on n

elements using composition as the group operation. (The sequence $s_1 s_2$ from the previous paragraph gave an example of the composition of two permutations).

Posets: A partially ordered set (poset) is a set P of objects equipped with a binary relation, typically denoted $\leq$, that is reflexive, antisymmetric, and transitive. This means that for all elements $a, b, c \in P$: (1) $a \leq a$, (2) If $a \leq b$ and $b \leq a$, then $b = a$, and (3) if $a \leq b$ and $b \leq c$, then $a \leq c$. Unlike total orders which are more familiar (e.g., $\mathbb{Z}$), in a partial order some pairs of elements may be incomparable. An example of a partially ordered set is the set of all subsets of $\{1,2,3,4\}$, ordered by inclusion. This is a partial order and not a total order because $\{1,2\}$ is not comparable to $\{2,3\}$ or to $\{2,3,4\}$, for example. In a poset, y covers x if y is greater than x with respect to the ordering, and there is no element z such that $y > z > x$. In this example, $\{1,2,4\}$ covers $\{1,2\}, \{2,4\}$, and $\{1,4\}$, but not $\{1\}, \{2\}$, or $\{4\}$.

2.0 Datasets

2.1 Partial orders on lattice paths (open problem)

[33] defines two order relations on NE lattice paths from $(0,0)$ to (a,b) called the *matching ordering* $(\leq_M)$ and the *Lagrange ordering* $(\leq_L)$, and proposes studying these partially ordered sets. The matching ordering assigns a number to each lattice path based on the number of perfect matchings of an associated snake graph, while the Lagrange ordering assigns a number to each lattice path equal to the Lagrange number of a continued fraction. These numbers each define the respective partial order. An open question related to the matching and Lagrange orders is whether we can find a simple way of determining whether two paths w and w' have the same relationship in both orders ($w \leq_L w'$ and $w \leq_M w'$) or different relationships in both orders ($w \leq_L w'$ and $w \geq_M w'$) or vice versa) [43].

Dataset: Pairs of NE 1 (w, w') on a grid of size $n \times n-1$ where w' covers w in either the matching or Lagrange order (but not both). We include $n = 10, 11, 12, 13$.

Task (classification): Train a model that can predict whether (w, w') is a covering pair in Lagrange or matching order.

2.2 Weaving patterns (open problem)

Weaving patterns are $(n \times n-1)$-matrices with $\{0,1\}$-entries introduced by Felsner [17] to study the number of reduced decompositions of the permutation $\sigma = n, n-1, \ldots, 1$ up to commutation equivalence. The number of such objects also counts the number of parallel sorting networks, the number of rhombic tilings of regular polygons, and is connected to the study of the higher Bruhat orders.

Weaving patterns can be enriched by replacing the $\{0,1\}$-entries to the matrix with $\{1,2,\ldots,n\}$-entries that track the element being swapped. An $O(n^2)$ algorithm for determining if a given 0-1 matrix is a valid weaving pattern exists but gives no additional insight into the structure of weaving patterns and correspondingly the asymptotics of reduced decompositions.

The enumeration of reduced decompositions up to commutation equivalence has been studied by many including Knuth and Stanley.

An exact formula is likely out of reach, so asymptotic upper and lower bounds are of great interest. ML models that can detect necessary or sufficient conditions for a matrix to be a valid weaving pattern have the potential to lead to substantial improvements in the upper bound.

Dataset: A mixture of enriched weaving patterns and non-weaving pattern matrices with $\{1, 2, \ldots, n\}$-entries.

Task: Classify whether a matrix in the dataset is a weaving pattern or not.

2.3 Mutation equivalence of quivers (open problem)

Quivers and quiver mutations are central to the combinatorial study of cluster algebras, algebraic structures with connections to Poisson Geometry, string theory, and Teichmuller theory. Leaving precise definitions for the Appendix, quivers are directed graphs, and a quiver

mutation is a local transformation of the graph involving certain vertices and arrows that produces a new quiver. A fundamental open problem in this area is to find an algorithm that determines whether two quivers are mutation equivalent. Currently, no such algorithm exists for quivers on more than three vertices.

Recent work has explored whether deep learning models can learn to correctly predict if two quivers are mutation equivalent [4]. Our dataset aims to facilitate continuation of this work. In [4] and in our dataset, quivers are represented using adjacency matrices.

Dataset: Adjacency matrices for seven quivers, each with 11 vertices, labeled by mutation equivalence class.

Task: Classify which mutation equivalence class an adjacency matrix corresponds to.

2.4 Symmetric group characters (classic result)

One way to understand the algebraic structure of permutations (symmetric groups, S_n) is through their representation theory [31], which converts algebraic questions into linear algebra questions that are often easier to solve.

A *representation* of group G on vector space V, is a map $\phi: G \to GL(V)$ converts elements of g to invertible matrices on vector space V and which respects the compositional structure of the group. A basic result in representation theory says that all representations of a finite group can be decomposed into atomic building blocks called *irreducible representations*. Amazingly, irreducible representations are themselves uniquely determined by the value of the trace, $Tr(\phi(g))$, where g ranges over subsets of G called conjugacy classes. These values are called *characters*.

The representation theory of symmetric groups has rich combinatorial interpretations. Both irreducible representations of S_n and the conjugacy classes of S_n are indexed by partitions of n and thus the characters of irreducible representations of S_n are indexed by two partitions of n. For $\lambda, \mu \vdash n$ we write χ_μ^λ. This combinatorial connection is not superficial, there are algorithms (e.g., the Murnaghan-Nakayama rule), which allow calculation of irreducible characters via simple manipulation of the Young diagrams for λ and μ without any reference to algebra.

Dataset: Pairs of integer partitions of n, λ, μ and the corresponding symmetric group character χ_μ^λ.

Task: Given partitions λ and μ, predict the irreducible symmetric group character χ_μ^λ.

2.5 The coefficients of Kazhdan-Lusztig polynomials (open problem)

Kazhdan-Lusztig (KL) polynomials are integer polynomials in a variable q that (for our purposes) are indexed by a pair of permutations [23]. We will write the KL polynomial associated with permutations σ and ν as $P_{\sigma,\nu}(q)$. For example, the KL polynomial associated with permutations $\sigma = 1\ 4\ 3\ 2\ 7\ 6\ 5\ 10\ 9\ 8\ 11$ and $\nu = 4\ 6\ 7\ 8\ 9\ 10\ 1\ 11\ 2\ 3\ 5$ is

$$P_{\sigma,\nu} = 1 + 16q + 103q^2 + 337q^3 + 566q^4 + 529q^5 + 275q^6 + 66q^7 + 3q^8$$

(this example comes from [1]). KL polynomials have deep connections throughout several areas of mathematics. For example, KL polynomials are related to the dimensions of intersection homology in Schubert calculus, the study of the Hecke algebra, and representation theory of the symmetric group. They can be computed via a recursive formula [23], nevertheless, in many ways they remain mysterious. For instance, there is no known closed formula for the degree of $P_{\sigma,\nu}(q)$.

One type of question of special interest is the value of coefficient on the terms of q in $P_{\sigma,\nu}(q)$. Perhaps most well-known is the question of the coefficient on term $q^{\ell(\sigma)-\ell(\nu)-1/2}$ (where $\ell(x)$ is a statistic called the length of the permutation), which is known as the μ-coefficient. Better understanding of this and other coefficients has the potential to shed considerable light on other aspects of this family of polynomials.

Dataset: Each instance in this dataset consists of a pair of permutations on n, σ and ν, along with the coefficients of $P_{\sigma,\nu}(q)$. We provide $n = 8,9,10$.

Task: The task to predict the coefficients of $P_{\sigma,\nu}(q)$ given σ and ν.

2.6 The mHeight function of a permutation (intermediary result)

The mHeight function is a statistic associated with a permutation that relates to all 3412-patterns in the permutation (see Section 3 for the definition of a 3412-pattern). It plays a crucial role in the proof by Gaetz and Gao [18] which resolved a long-standing conjecture of Billey and Postnikov [8] about the coefficients on Kazhdan-Lusztig polynomials which carry important geometric information about certain spaces, Schubert varieties, that are of interest both to mathematicians and physicists. The task of predicting the mHeight function thus represents an interesting opportunity to understand whether a non-trivial intermediate step in an important proof can be learned by machine learning.

Let $\sigma = a_1 \dots a_n \in S_n$ be a permutation containing at least one occurrence of a 3412 pattern. Let (a_i, a_j, a_k, a_ℓ) be a 3412 pattern so that $1 \le i < j < \ell \le n$ but $a_k < a_\ell < a_i < a_j$. The *height* of (a_i, a_j, a_k, a_ℓ) is $a_\ell - a_i$. The *mHeight* of σ is then the minimum height over all 3412 patterns in σ.

Dataset: Permutations of size n labeled by their mHeight. We provide datasets for $n = 10, 11, 12$.

Task: Predict the mHeight of a permutation.

2.7 Schubert polynomial structure constants (open problem)

Schubert polynomials [7, 15, 25] are a family of polynomials indexed by permutations of S_n. Developed to study the cohomology ring of the flag variety, they have deep connections to algebraic geometry, Lie theory, and representation theory. Despite their geometric origins, Schubert polynomials can be described completely combinatorially ([9,6]), making them a well-studied object in algebraic combinatorics. An important open problem in the study of Schubert polynomials is understanding their *structure constants*.

When two Schubert polynomials are multiplied, their product is a linear combination of Schubert polynomials, i.e. $\mathcal{S}_\beta \mathcal{S}_\gamma = \sum_\alpha c_{\beta\gamma}^\alpha \mathcal{S}_\alpha$. The question is whether the $c_{\beta\gamma}^\alpha$ (the structure constants)

have a combinatorial description or formula. To give an example of what we mean by combinatorial description, in the case of Schur polynomials (which can be viewed as specific case of Schubert polynomials), the coefficients in the product are equal to the number of semistandard tableaux satisfying certain properties.

Dataset: Each instance in this dataset is a triple of permutations (β, γ, α), labeled by its coefficient $c_{\beta\gamma}^{\alpha}$ in the expansion of the product $\mathcal{S}_\beta \mathcal{S}_\gamma$. Not all possible triples of permutations are included; the dataset consists of an approximately equal number of zero and nonzero coefficients. We provide datasets for $n = 4,5,6$.

Task: The task is to predict the coefficient $c_{\beta\gamma}^{\alpha}$.

2.8 Robinson-Schensted-Knuth correspondence (classic result)

The Robinson-Schensted (RS) algorithm [30,32] gives a bijection between pairs of standard Young tableau of the same shape $\lambda \vdash n$ and permutations in S_n of conjugacy class λ, providing a bijective proof of a fundamental identity from representation theory. Knuth extended the RS algorithm to a bijection known as the Robinson-Schensted-Knuth (RSK) correspondence, which maps matrices of non-negative integers to pairs of semistandard Young tableaux of the same shape. This correspondence is significant in algebraic combinatorics not only because of the connection it provides between the combinatorial structure of Young tableaux and the theory of symmetric functions, but also because of the many generalizations and variants it has inspired, which has led to substantial progress in the field.

The goal of this benchmark is to see whether a model can learn the RSK algorithm. That is, for a fixed n the model is provided with a permutation $\pi \in S_n$ and required to predict pairs of standard Young tableaux. Although the algorithm is known, it would be significant for a model to learn this correspondence due to the the intricate combinatorial rules involved. Notably, the RSK correspondence can be used to find the length of the longest increasing subsequence, so a model that learns this algorithm implicitly must also learn to solve the increasing subsequence problem. Additionally, given the numerous generalizations of the RSK correspondence, a model that performs well on this benchmark could potentially be investigated for its ability to generalize to other related combinatorial settings.

Dataset: This dataset consists of triples: two standard Young tableau of size n and their corresponding permutation (via the RSK algorithm). We include datasets for $n = 8, 9, 10$.

Task: Given pairs of standard Young tableau, predict the corresponding permutation.

2.9 Grassmannian Cluster Algebras and SSYTs (open problem)

The Grassmann manifold $Gr(k, n)$ is the set of full-rank $k \times n$ matrices up to equivalence of elementary row operations (equivalently the space whose points are k-dimensional subspaces in $\mathbb{R}^n$). Grassmannians are of fundamental geometric importance and are a central tool in a model of quantum field theory known as supersymmetric Yang-Mills theory.

Among the many algebraic-combinatorial properties of Grassmannians is an algebraic structure on its coordinate ring making it a cluster algebra. A recent result of Chang, Duan, Fraser, and Li [11] parameterize cluster variables of the Grassmannian coordinate ring in terms of equivalence classes of semistandard Young tableaux. Not every semistandard Young tableaux indexes a

cluster variable and a natural question to ask is which are valid cluster variable indices. A necessary condition is that the tableaux is of rectangular shape. We follow the set-up set up of [12] who first applied machine learning to this problem, though we choose a different method of sampling tableau that do not index cluster variables.

Dataset: A collection of rectangular semistandard Young tableau each with a label indicating whether they index a cluster variable or not.

Task: Predict whether a Young tableaux indexes a cluster variable.

3.0 Discussion

3.1 Challenges when generating mathematics benchmarks

Generating useful benchmark datasets for mathematics problems presents a number of challenges. Some of these are common to many scientific domains while others are more specific to mathematics. In the former case, imbalance is an issue in several ways. On the one hand, traditional class imbalance comes up frequently. For instance, if the symmetric group character training dataset for $n = 18$ is treated as a classification problem, it has $46{,}285$ instances where the character value is 0 and only two instances where the character value is $16{,}336{,}320$.

Besides class imbalance, there can also be imbalance in terms of how interesting examples of a dataset are. For a given task, it may be the case that the vast majority of randomly sampled instances are uninteresting because they can be predicted or classified for straightforward reasons. In these cases, individual instances do not capture the mathematics that we care about. When enough data exists, one way to mitigate this situation is to subsample for harder examples. This is what we did for a number of the datasets in ACBench including Weaving Patterns where we imposed some additional constraints on the non-weaving pattern $\{1,2,\dots,n\}$-matrices to make them harder to distinguish from true weaving patterns. Similarly, after training on our initial version of the Grassmannian Cluster Algebras dataset (created by [12]), interpretability methods showed that our models (which performed very well) had simply identified that one can't have a 3 in the top right and a 10 in the bottom left. We revisited our negative sampling algorithm to create a problem where a model would be forced to learn features that were more interesting.

Finally, the choice of input representation can have large downstream impacts on how hard it is for a model to learn to solve a task. For example, there are many equivalent ways to represent a permutation. Similar to other parity prediction tasks [20], prediction of permutation parity is a hard task for transformers. However, these models do substantially better when input permutations are represented via their inversion vector rather than one-line notation. At an intuitive level, this feels unsurprising since the process of calculating parity is much easier if one already has the inversion set. Thus, one can often change the difficulty of a problem (with a known solution) by optimizing the input representation. Since one of the goals of this benchmark is to try to help develop ML methods that can be applied to open problems where we do not know the best representation, we have usually opted to represent datatypes in a standard way.

Dataset	Logistic regression	Transformer
Lattice paths		
$n = 9$	66.2% ± 0.00%	66.2 % ± 0.00%
$n = 10$	66.3% ± 0.00%	66.3 % ± 0.00%
$n = 11$	66.5% ± 0.00%	33.5% ± 0.00%
$n = 12$	75% ± 0.00%	33.4% ± 0.00%
Weaving patterns		
$n = 6$	70.4% ± 0.00%	88.7% ± 0.01%
$n = 7$	85.8%± 0.00%	99.1% ± 0.00%
Cluster algebra quivers	40.3 %± 0.00%	43.2%± 0.00%
S_n characters		
$n = 18$	40.6%± 0.00%	42.19 %± 0.00%
$n = 20$	40.64%± 0.00%	41.92 %± 0.00%
$n = 22$	41.01%± 0.00%	41.61 %± 0.00%
Grassmanian cluster algebras		
$n = 6$	65.7%± 0.00%	50.0 %± 0.00%
KL polynomials		
$n = 8$	78.2%± 0.00%	84.8% ± 0.00%
$n = 9$	57.0%± 0.00%	81.8% ± 0.01%
Schubert polynomials		
$n = 3$	76.5%± 0.00%	72.9% ± 0.03%
$n = 4$	64.4%± 0.00%	97.7% ± 0.00%
$n = 5$	66.7%± 0.00%	97.3% ± 0.00%
mHeight		
$n = 8$	64.6%± 0.00%	72.8% ± 0.01%
$n = 9$	70.8%± 0.00%	83.7% ± 0.00%
$n = 10$	94.2%± 0.00%	94.2% ± 0.00%

Figure 3: Baseline performance of two model types: logistic regression and transformers

3.2 Baselines

We ran our initial baselines on all datasets in the benchmark using transformers and logistic regression. Our results are summarized in Figure 1. As can be seen, the performance of models varies considerably on different tasks. Some tasks are quite hard (such as symmetric group character calculation) with results that are nearly equivalent to guessing the most populous class. Other tasks were considerably easier such as predicting the mHeight function of a permutation.

3.3 Dependence on n

Many problems in algebraic combinatorics have a natural dependence on a parameter n. We have chosen to structure datasets in ACBench to reflect this, with the majority of datasets taking the form of a series of datasets $\{D_n\}_{n\geq 1}$. We provide a few values of n and, in many cases, the code to generate others.

There is no reason one could not combine all D_n into a single large dataset, but we decided that retaining the dependence on n provided an interesting additional parameter that could be used to probe ML algorithms. For instance, generalization from D_{n-1} to D_n or changes in model performance as n grows.

Generally, there are two properties that change as $n \to \infty$. First, the size of D_n grows as n grows. The rate of growth depends on the specific problem, with many $|D_n|$ growing exponentially (such as those datasets that depend on the number of permutations of n). On the other hand, the problems also tend to become more complex. We were curious of how this

balance would play out empirically when we trained models on different D_n. Experimentally we found mixed results. For example, we ran 5 2-layer MLP models for 500 epochs on the *Lattice Path Datasets* corresponding to grids of size $6 \times 5, \ldots, 13 \times 12$. We see in Figure 4 (left) that with an interesting exception of moving from 7×6 to 8×7, performance across a range of dimensions improves as n grows.

On the other hand, we also looked at sampling from greater depth when exploring the Mutation Equivalent Quiver dataset (this means allowing a greater number of mutations to be applied to the initial quiver). As shown in Figure 4 (center), we find that performance somewhat degrades even though the size of the datasets increases. We suspect that exploration of the complexity of these problems (where it is known) might be an avenue for shedding light on this phenomenon. In either case, it seems safe to say that ML algorithms perform best at a scale which human mathematicians would likely have trouble fully absorbing (an algorithm can learn from billions of permutations, but humans are more limited in what we can hope to look over).

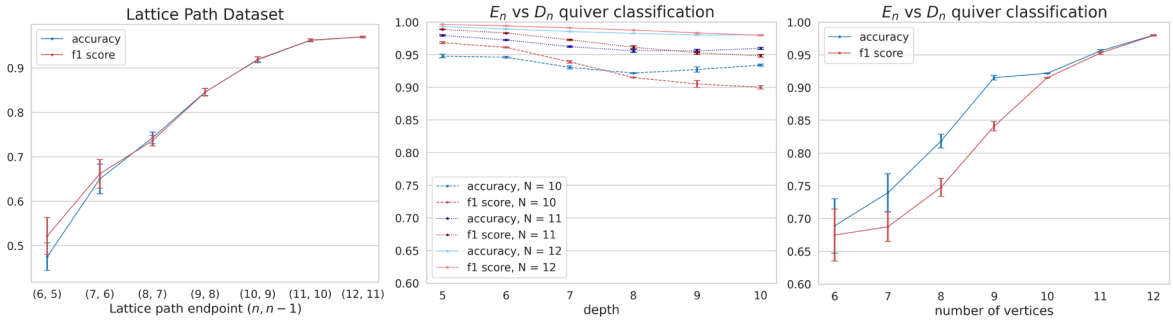

Figure 4: **(Left)** Performance on the *Lattice Path Dataset* as a function of the width of the $n \times n-1$ grid on which lattice paths are constrained to. As n grows in $n \times n-1$, the training set size increases but problem complexity may also grow. **(Center)** Performance on the type E versus type D quiver classification task as a function of the depth, which must be specified for type E quivers on $N = 10, 11, 12$ vertices, and **(Right)** the number of vertices N.

4.0 Conclusion and Limitations

In this paper we introduced Algebraic Combinatorics Benchmarks (ACBench), a collection of datasets structured for machine learning and designed to facilitate the development of machine learning methods for advancing research level mathematics. While we believe that these datasets will provide significant value to the ML community, they also have some limitations. Firstly, even within the field of algebraic combinatorics (which is just one subfield of mathematics) they only represent a small slice of the total breadth of problems. We hope that community feedback will help us to fill some of these gaps. Also, a range of choices needed to be made when structuring these datasets (such as choosing the representation of a datatype or sampling strategy). While we think we made reasonable choices, the novelty of the field of AI for math means that we can't be certain of this. Despite these limitations, we believe that ML tools for mathematics is a promising route to a richer and more diverse mathematics. We hope that these benchmarks will be useful to researchers looking to make progress in this area.

5.0 References

1. Klmu & klc - versions 1.0. https://gswarrin.w3.uvm.edu/research/klc/klc.html. Accessed: 2024-06-05.

2. Ainsworth, S. K., Hayase, J., & Srinivasa, S. (2022). Git re-basin: Merging models modulo permutation symmetries. *arXiv preprint arXiv:2209.04836*.

3. Azerbayev, Z., Schoelkopf, H., Paster, K., Dos Santos, M., McAleer, S., Jiang, A. Q., Deng, J., Biderman, S., & Welleck, S. (2023). Llemma: An open language model for mathematics. *arXiv preprint arXiv:2310.10631*.

4. Bao, J., Franco, S., He, Y.-H., Hirst, E., Musiker, G., & Xiao, Y. (2020). Quiver mutations, Seiberg duality, and machine learning. *Physical Review D*, 102(8), 086013.

5. Bao, J., He, Y.-H., Heyes, E., & Hirst, E. (2022). Machine learning algebraic geometry for physics. *arXiv preprint arXiv:2204.10334*.

6. Bergeron, N., & Billey, S. (1993). RC-graphs and schubert polynomials. *Experimental Mathematics*, 2(4), 257-269.

7. Bernstein, I. M. G. I. N., Gel'fand, I. M., & Gel'fand, S. I. (1973). Schubert cells and cohomology of the spaces g/p. *Russian Mathematical Surveys*, 28(3), 1.

8. Billey, S., & Postnikov, A. (2005). Smoothness of Schubert varieties via patterns in root subsystems. *Adv. in Appl. Math.*, 34(3), 447-466.

9. Billey, S. C., Jockusch, W., & Stanley, R. P. (1993). Some combinatorial properties of Schubert polynomials. *Journal of Algebraic Combinatorics*, 2(4), 345-374.

10. Butter, A., Plehn, T., Schumann, S., Badger, S., Caron, S., Cranmer, K., Di Bello, F. A., Dreyer, E., Forte, S., Ganguly, S., et al. (2023). Machine learning and LHC event generation. *SciPost Physics*, 14(4), 079.

11. Chang, W., Duan, B., Fraser, C., & Li, J.-R. (2020). Quantum affine algebras and Grassmannians. *Math. Z.*, 296(3-4), 1539-1583.

12. Cheung, Man-Wai, et al. "Clustering cluster algebras with clusters." *arXiv preprint arXiv:2212.09771* (2022).

13. Coates, T., Kasprzyk, A., & Veneziale, S. (2024). Machine learning detects terminal singularities. *Advances in Neural Information Processing Systems*, 36.

14. Davies, A., Veličković, P., Buesing, L., Blackwell, S., Zheng, D., Tomašev, N., Tanburn, R., Battaglia, P., Blundell, C., Juhász, A., et al. (2021). Advancing mathematics by guiding human intuition with AI. *Nature*, 600(7887), 70-74.

15. Demazure, M. (1974). Desingularisation des varietes de Schubert generalises. In *Annales scientifiques de l'Ecole Normale Superieure* (Vol. 7, pp. 53-88).

16. Entezari, R., Sedghi, H., Saukh, O., & Neyshabur, B. (2021). The role of permutation invariance in linear mode connectivity of neural networks. *arXiv preprint arXiv:2110.06296*.

17. Felsner, S. (1997). On the number of arrangements of pseudolines. In *ACM Symposium on Computational Geometry (Philadelphia, PA, 1996)* (Vol. 18, pp. 257-267).

18. Gaetz, Christian, and Yibo Gao. On the minimal power of q in a Kazhdan–Lusztig polynomial" *Advances in Mathematics* 457 (2024): 109941.

19. Godfrey, C., Brown, D., Emerson, T., & Kvinge, H. (2022). On the symmetries of deep learning models and their internal representations. *Advances in Neural Information Processing Systems*, 35, 11893-11905.

20. Hahn, M., & Rofin, M. (2024). Why are sensitive functions hard for transformers? *arXiv preprint arXiv:2402.09963*.

21. Jumper, J., Evans, R., Pritzel, A., Green, T., Figurnov, M., Ronneberger, O., Tunyasuvunakool, K., Bates, R., Žídek, A., Potapenko, A., et al. (2021). Highly accurate protein structure prediction with AlphaFold. *Nature*, 596(7873), 583-589.

22. Kazalicki, M., & Vlah, D. (2023). Ranks of elliptic curves and deep neural networks. *Research in Number Theory*, 9(3), 53.

23. Kazhdan, D., & Lusztig, G. (1979). Representations of Coxeter groups and Hecke algebras. *Inventiones mathematicae*, 53(2), 165-184.

24. Keriven, N., & Peyré, G. (2019). Universal invariant and equivariant graph neural networks. *Advances in Neural Information Processing Systems*, 32.

25. Lascoux, A., & Schützenberger, M.-P. (1982). Structure de Hopf de l'anneau de cohomologie et de l'anneau de grothendieck d'une variete de drapeaux. *CR Acad. Sci. Paris Ser. I Math*, 295(11), 629-633.

26. Lee, J., Lee, Y., Kim, J., Kosiorek, A., Choi, S., & Teh, Y. W. (2019). Set transformer: A framework for attention-based permutation-invariant neural networks. In *International conference on machine learning* (pp. 3744-3753). PMLR.

27. Lee, N., Sreenivasan, K., Lee, J. D., Lee, K., & Papailiopoulos, D. (2023). Teaching arithmetic to small transformers. *arXiv preprint arXiv:2307.03381*.

28. Liu, Z., Gan, E., & Tegmark, M. (2023). Seeing is believing: Brain-inspired modular training for mechanistic interpretability. *Entropy*, 26(1), 41.

29. Nanda, Neel, et al. "Progress measures for grokking via mechanistic interpretability." *arXiv preprint arXiv:2301.05217* (2023).

30. G de B Robinson. (1938). On the representations of the symmetric group. *American Journal of Mathematics* (pp. 745-76.

31. Sagan, B. (2013). The symmetric group: representations, combinatorial algorithms, and symmetric functions. *Springer Science & Business Media*. volume 203.

32. Schensted, C. (1961). Longest increasing and decreasing subsequences. *Canadian Journal of mathematics*, 13:179—191.

33. Schiffler, R. (2023). Perfect matching problems in cluster algebras and number theory. *arXiv preprint arXiv:2302.02185.*

34. Song, P., Yang, K., Anandkumar, A. (2024). Towards large language models as copilots for theorem proving in Lean. *arXiv preprint arXiv:2404.12534*.

35. Stanley, R. (2011). Enumerative combinatorics volume 1 second edition. *Cambridge studies in advanced mathematics*.

36. Stanley, R. (2011). Enumerative combinatorics volume 2 second edition. *Cambridge studies in advanced mathematics*.

37. Stein, W. (2024). Sage Mathematics Software. The Sage Development Team.

38. Velickovic, P., Badia, A., Budden, D., Pascanu, R., Banino, A., Dashevskiy, M., Hadsell, Raia, Blundell, C., (2022). The CLRS algorithmic reasoning benchmark. *International Conference on Machine Learning*, pp. 22084-22102. PMLR.

39. Velickovic, P. (2023). Neural algorithmic reasoning. *The Gradient.*

40. Wagner, A. (2021). Constructions in combinatorics via neural networks. *arXiv preprint arXiv:2104.14516*.

41. Yang, K., Swope, A., Gu, A., Chalamala, R., Song, P., Yu S., Godil, S., Prenger, R., Anandkumar, A. (2024). Leandojo: Theorem proving with retrieval-augmented language models. *Advances in Neural Information Processing Systems, 36*.

42. Zaheer, M., Kottur, S., Ravanbakhsh, S., Poczos, B., Salakhutdinov, R., Smola, A. (2017). Deep sets. *Advances in neural information processing systems, 30*.

43. Zhong, Z., Liu, Z., Tegmark, M., Andreas, J. (2024). The clock and the pizza: Two stories in mechanistic explanation of neural networks. *Advances in Neural Information Processing Systems, 36.*

44. Apruzzese P., Cong, K. (2023). On two orderings of lattice paths. *arXiv preprint arXiv:2310.16963*.

Pacific Northwest National Laboratory

902 Battelle Boulevard
P.O. Box 999
Richland, WA 99354

1-888-375-PNNL (7665)

www.pnnl.gov