ADV. THEOR. MATH. PHYS.
Volume 27, Number 3, 797–828, 2023

Clustering cluster algebras with clusters

Man-Wai Cheung, Pierre-Philippe Dechant, Yang-Hui He,
Elli Heyes, Edward Hirst, and Jian-Rong Li

Classification of cluster variables in cluster algebras (in particular, Grassmannian cluster algebras) is an important problem, which has direct applications to computations of scattering amplitudes in physics. In this paper, we apply the tableaux method to classify cluster variables in Grassmannian cluster algebras $\mathbb{C}[\mathrm{Gr}(k, n)]$ up to $(k, n) = (3, 12), (4, 10)$, or $(4, 12)$ up to a certain number of columns of tableaux, using HPC clusters. These datasets are made available on GitHub. Supervised and unsupervised machine learning methods are used to analyse this data and identify structures associated to tableaux corresponding to cluster variables. Conjectures are raised associated to the enumeration of tableaux at each rank and the tableaux structure which creates a cluster variable, with the aid of machine learning.

1. Introduction

Cluster algebras were first introduced by Fomin and Zelevinsky in 2000 [21] in the context of Lie theory but have since been applied to many other areas of mathematics and physics, such as in integrable systems, tropical geometry, and scattering amplitudes. Classification of cluster variables (in particular cluster variables in Grassmannian cluster algebras) is important in mathematics [30, 33] and scattering amplitudes in physics [3, 4, 19, 20, 22–25, 28].

For example, in mathematics, cluster variables in Grassmannian cluster algebras $\mathbb{C}[\mathrm{Gr}(k,n)]$ correspond to real prime modules of the quantum affine algebra $U_q(\widehat{\mathfrak{sl}_k})$, [13, 30]. They also correspond to rigid indecomposable modules in Grassmannian cluster categories [9]. In physics, some particular scattering amplitudes in $N=4$ super Yang-Mills theory can be written as sums of polylogarithms in variables with a cluster algebra structure, [25]. Remainder functions of MHV scattering amplitudes in the planar limit of $N=4$ super Yang-Mills theory tend to be linear combinations of generalized polylogarithms whose symbols are composed of X-coordinates of the the cluster algebra $\mathbb{C}[\mathrm{Gr}(4,n)]$, [25]. Cluster X-coordinates can be obtained from cluster A-coordinates. In this paper, cluster A-coordinates are called cluster variables.

In short, a cluster algebra is a commutative ring generated inside an ambient field. It is defined iteratively from an initial seed, consisting of a set of variables, called a cluster, and a quiver, via a procedure called mutation. The mutation process produces further seeds, which consist of clusters and quivers. The cluster algebra is the algebra generated by all cluster variables (including frozen variables); see §2.1 for more details.

As a set, for integers $k \le n$, the Grassmannian variety $\mathrm{Gr}(k,n)$ is the set of all k-dimensional subspaces of the n dimensional vector space $\mathbb{C}^n$. Scott [40] proved that there is a cluster algebra structure in the coordinate ring $\mathbb{C}[\mathrm{Gr}(k,n)]$. The ring $\mathbb{C}[\mathrm{Gr}(k,n)]$ is called a Grassmannian cluster algebra.

Hernandez and Leclerc [30] showed that there is a cluster algebra structure on the Grothendieck ring $K_0(\mathcal{C}_\ell)$ of a certain subcategory $\mathcal{C}_\ell$ of the category of finite-dimensional modules of a quantum affine algebra $U_q(\widehat{\mathfrak{g}})$. In the case when $\mathfrak{g}=\mathfrak{sl}_k$, the cluster algebra $K_0(\mathcal{C}_\ell)$ is isomorphic to the cluster algebra $\mathbb{C}[\mathrm{Gr}(k,n,\sim)]$, where $\mathbb{C}[\mathrm{Gr}(k,n,\sim)]$ is the quotient of the Grassmannian cluster algebra $\mathbb{C}[\mathrm{Gr}(k,n)]$ by identifying certain frozen variables with 1, cf. [30, 40].

Simple modules of $U_q(\widehat{\mathfrak{g}})$ are parametrized by dominant monomials in formal variables $Y_{i,s}$, $i \in I$, $s \in \mathbb{Z}$, where I is the set of vertices of the Dynkin diagram of $\mathfrak{g}$, cf. [15]. It is shown in [13] that, in the case of $\mathfrak{g}=\mathfrak{sl}_k$, the monoid of dominant monomials is isomorphic to the monoid $\mathrm{SSYT}(k,[n],\sim)$, where $\mathrm{SSYT}(k,[n],\sim)$ is a quotient of the monoid $\mathrm{SSYT}(k,[n])$ of semistandard tableaux (SSYT) of rectangular shape with k rows and with entries in $[n] = \{1, \ldots, n\}$, cf. §2.2. Therefore, every simple module of $U_q(\widehat{\mathfrak{sl}_k})$ corresponds to a semistandard Young tableau in $\mathrm{SSYT}(k, [n], \sim)$. It follows, that the dual canonical basis of $\mathbb{C}[\mathrm{Gr}(k, n, \sim)]$ is in one-to-one correspondence with tableaux in $\mathrm{SSYT}(k, [n], \sim)$. In particular, cluster variables in $\mathbb{C}[\mathrm{Gr}(k, n, \sim)]$ correspond to a tableau in $\mathrm{SSYT}(k, [n], \sim)$.

The set of cluster variables in $\mathbb{C}[\mathrm{Gr}(k, n)]$ is the union of the set of cluster variables in $\mathbb{C}[\mathrm{Gr}(k, n, \sim)]$ and the frozen variables in $\mathbb{C}[\mathrm{Gr}(k, n)]$. Therefore, in order to classify cluster variables in $\mathbb{C}[\mathrm{Gr}(k, n)]$, it suffices to classify cluster variables in $\mathbb{C}[\mathrm{Gr}(k, n, \sim)]$. We say that a tableau in $\mathrm{SSYT}(k, [n], \sim)$ (resp. $\mathrm{SSYT}(k, [n])$) is a cluster variable if the dual canonical basis element corresponding to it is a cluster variable in $\mathbb{C}[\mathrm{Gr}(k, n, \sim)]$ (resp. $\mathbb{C}[\mathrm{Gr}(k, n)]$). Up to frozen variables, cluster variables in $\mathrm{SSYT}(k, [n], \sim)$ and $\mathrm{SSYT}(k, [n])$ are the same.

A simple $U_q(\widehat{\mathfrak{g}})$-module M is called real if $M \otimes M$ is simple, cf. [35]. A simple $U_q(\widehat{\mathfrak{g}})$-module M is called prime if M is not isomorphic to $M_1 \otimes M_2$ for any non-trivial modules M_1, M_2, cf. [14]. Hernandez and Leclerc [30] conjectured that real prime modules of $U_q(\widehat{\mathfrak{g}})$ are in one-to-one correspondence to cluster variables in $K_0(\mathcal{C}_\ell)$. Therefore it is important to classify cluster variables in $K_0(\mathcal{C}_\ell)$, and equivalently in $\mathbb{C}[\mathrm{Gr}(k, n)]$.

In the context of planar $N = 4$ super Yang-Mills theory, the cluster variables in $\mathbb{C}[\mathrm{Gr}(k, n)]$ appear as symbol letters of scattering amplitudes, [4, 19, 20, 25, 28, 29]. Therefore classification of cluster variables in $\mathbb{C}[\mathrm{Gr}(k, n)]$ is also important in physics.

For $T \in \mathrm{SSYT}(k, [n])$, we say that T is of rank d if T has d columns. We say that a cluster monomial (in particular, a cluster variable) is of rank r if the corresponding tableau has rank r. For general $k \leq n$, the number of cluster variables in $\mathbb{C}[\mathrm{Gr}(k, n)]$ is infinite. On the other hand, if we count cluster variables in $\mathbb{C}[\mathrm{Gr}(k, n)]$ with a given rank, then the number is finite. This is because the number of semistandard Young tableaux with a given rank is finite.

In this paper, we apply high-performance computing (HPC) clusters to compute cluster variables in $\mathbb{C}[\mathrm{Gr}(k, n)]$. Our method of computing cluster variables is to use mutation of tableaux introduced in [13], cf. Formula (2). We compute the cluster variables in the Grassmannian cluster algebras $\mathbb{C}[\mathrm{Gr}(3, 12)]$ up to rank 6, $\mathbb{C}[\mathrm{Gr}(4, 12)]$ up to rank 4, and $\mathbb{C}[\mathrm{Gr}(4, 10)]$ up to rank 6, cf. Table 1. The datasets produced amount to ~ 0.75Gb of data and took ~ 0.5 million core hours to compute.

From our results, we obtain conjectural formulas for numbers of cluster variables of certain ranks in $\mathbb{C}[\mathrm{Gr}(3,n)]$ and $\mathbb{C}[\mathrm{Gr}(4,n)]$, cf. Conjecture 3.1:

$$\begin{aligned}
N_{3,n,3} &= 24\binom{n}{8} + 9\binom{n}{9}, \\
N_{3,n,4} &= 288\binom{n}{9} + 400\binom{n}{10} + 264\binom{n}{11} + 48\binom{n}{12}, \\
N_{4,n,3} &= 174\binom{n}{8} + 855\binom{n}{9} + 1285\binom{n}{10} + 693\binom{n}{11} + 123\binom{n}{12},
\end{aligned}$$

where $N_{k,n,r}$ is the number of cluster variables of rank r in $\mathbb{C}[\mathrm{Gr}(k,n)]$.

We also conjecture that when one replaces a set of numbers $a_1 < \ldots < a_m$ appearing in a cluster variable (tableau) with another set of numbers $a'_1 < \ldots < a'_m$, one will obtain another cluster variable, cf. Conjecture 3.2.

Grassmannian cluster algebras have many cluster variables, forming large datasets with rich structure. Therefore, in addition to computing this data and making it readily available for physical and mathematical application, we also turn to techniques from data science and machine learning (ML) to analyse these datasets of variables and extract some of this structure.

More specifically, we would like to study the following problems:

Problem 1.1. *Can machine learning methods identify whether a given semistandard Young tableau corresponds to a cluster variable?*

Problem 1.2. *What structure of these tableaux can be extracted by machine learning techniques which identifies the tableau as corresponding to a cluster variable?*

The machine learning methods we employ include both supervised and unsupervised methods. Support Vector Machines and Neural Networks both learn to distinguish – with strong performance – tableaux from different algebras, and also learn to distinguish those tableaux that are cluster variables from those that are not. Principal Component Analysis and K-Means Clustering, also highlight to us the key features in the tableau data.

This paper is structured as follows: In §2 we establish the relevant mathematical background surrounding Grassmannian cluster algebra cluster variables and their representation as semistandard Young tableaux. In §3 we provide information regarding the computation of cluster variables in Grassmannian cluster algebras. In §4 we analyse the generated data using techniques from supervised and unsupervised machine learning. Conclusions are presented in §5.

Coding scripts, and data used in this work are available at the respective GitHub repository: `https://github.com/edhirst/GrassmanniansML.git`

2. Grassmannian cluster algebras

In this section, we recall results in [13, 21, 40] about cluster algebras and Grassmannian cluster algebras.

2.1. Cluster algebras

We begin by recalling the definition of cluster algebras given by Fomin and Zelevinsky [21].

A quiver $Q = (Q_0, Q_1, s, t)$ is a directed graph without loops or 2-cycles that can be described by a vertex set Q_0, an arrow set Q_1, and maps $s, t : Q_1 \to Q_0$ that take an arrow to its source and target, respectively. We identify $Q_0 = [m] = \{1, \ldots, m\}$ and declare vertices $1, \ldots, r$ as mutable vertices and vertices $r+1, \ldots, m$ as frozen vertices.

For $k \in [r]$, the mutated quiver $\mu_k(Q)$ is a quiver obtained from Q by:

(i) for each sub-quiver $i \to k \to j$, add a new arrow $i \to j$,

(ii) reverse the orientation of every arrow with target or source equal to k,

(iii) remove the arrows in a maximal set of pairwise disjoint 2-cycles.

Let $\mathcal{F}$ be an ambient field abstractly isomorphic to a field of rational functions in m independent variables. A seed in $\mathcal{F}$ is a pair $(\mathbf{x}, Q)$, where $\mathbf{x} = (x_1, \ldots, x_m)$ is a free generating set of $\mathcal{F}$, called a cluster, and Q is a quiver. The variables $x_1, \ldots, x_r$ are called cluster variables, and the variables $x_{r+1}, \ldots, x_m$ are called frozen variables.

For a seed $(\mathbf{x}, Q)$ and $k \in [r]$, the mutated seed $\mu_k(\mathbf{x}, Q)$ is $(\mathbf{x}', \mu_k(Q))$, where $\mathbf{x}' = (x'_1, \ldots, x'_m)$ with $x'_j = x_j$ for $j \neq k$ and $x'_k \in \mathcal{F}$ determined by

$$x'_k x_k = \prod_{\alpha \in Q_1, s(\alpha)=k} x_{t(\alpha)} + \prod_{\alpha \in Q_1, t(\alpha)=k} x_{s(\alpha)}.$$

After making a choice of an initial labeled seed, we say that a seed is reachable if it can be obtained from the initial seed by a finite sequence of mutations. One defines the clusters (resp. cluster variables) to be the clusters (resp. cluster variables) appearing in all reachable seeds. Two cluster variables are called compatible if they appear together in a cluster. A cluster monomial is a product of compatible cluster variables. The cluster algebra is the $\mathbb{C}$-algebra generated by all cluster variables and frozen variables.

2.2. Grassmannian cluster algebras and semistandard Young tableaux

We denote by $\mathrm{Gr}(k,n)$ the Grassmannian of k-planes in $\mathbb{C}^n$ and $\mathbb{C}[\mathrm{Gr}(k,n)]$ its homogeneous coordinate ring. It was shown by Scott [40] that the ring $\mathbb{C}[\mathrm{Gr}(k,n)]$ has a cluster algebra structure. Furthermore, it was shown in [13] that every cluster monomial (in particular, every cluster variable) in $\mathbb{C}[\mathrm{Gr}(k,n)]$ corresponds to a semistandard Young tableau. This was achieved by using the isomorphism between two cluster algebras: one cluster algebra is the Grothendieck ring of a certain subcategory of the category of finite-dimensional modules of the quantum affine algebra $U_q(\widehat{\mathfrak{sl}_k})$; the other cluster algebra is $\mathbb{C}[\mathrm{Gr}(k,n,\sim)]$, where $\mathbb{C}[\mathrm{Gr}(k,n,\sim)]$ is the quotient of $\mathbb{C}[\mathrm{Gr}(k,n)]$ by the ideal $\langle P_{i,i+1,\ldots,i+k-1}-1, \quad i\in[n-k+1]\rangle$.

For $k\le n$, we denote by $\mathrm{SSYT}(k,[n])$ the set of all semistandard Young tableaux of rectangular shape with k rows and with entries in $[n]=\{1,\ldots,n\}$.

For $S,T\in\mathrm{SSYT}(k,[n])$, we denote by $S\cup T$ the row-increasing tableau whose ith row is the union of the ith rows of S and T (as multisets). It was shown in [13] that for any $S,T\in\mathrm{SSYT}(k,[n])$, $S\cup T$ is in $\mathrm{SSYT}(k,[n])$.

We call S a factor of T, and write $S\subset T$, if the ith row of S is contained in that of T (as multisets), for $i\in[k]$. In this case, we define $\frac{T}{S}=S^{-1}T=TS^{-1}$ to be the row-increasing tableau whose ith row is obtained by removing that of S from that of T (as multisets), for $i\in[k]$.

A tableau $T\in\mathrm{SSYT}(k,[n])$ is trivial if each entry of T is one less than the entry below it.

For any $T\in\mathrm{SSYT}(k,[n])$, we denote by $T_{\mathrm{red}}\subset T$ the semistandard tableau obtained by removing a maximal trivial factor from T. That is, T_{red} is the tableau with the minimal number of columns such that $T=T_{\mathrm{red}}\cup S$ for a trivial tableau S. For trivial T, one has $T_{\mathrm{red}}=\mathbb{1}$ (the empty tableau). For $S,T\in\mathrm{SSYT}(k,[n])$, we define $S\sim T$ if $S_{\mathrm{red}}=T_{\mathrm{red}}$. The reduction relation "$\sim$" is an equivalence relation. We denote by $\mathrm{SSYT}(k,[n],\sim)$ the set of $\sim$-equivalence classes.

We use the same notation for a tableau T and its equivalence class, writing either $T \in \mathrm{SSYT}(k,[n])$ or $T \in \mathrm{SSYT}(k,[n],\sim)$ when it is important to distinguish these.

Example 2.1. We illustrate the operations $\cup$ and $\sim$:

$$\begin{array}{|c|c|}\hline 1 & 3 \\ \hline 2 & 7 \\ \hline 6 & 11 \\ \hline \end{array} \cup \begin{array}{|c|c|}\hline 1 & 7 \\ \hline 2 & 9 \\ \hline 8 & 10 \\ \hline \end{array} = \begin{array}{|c|c|c|c|}\hline 1 & 1 & 3 & 7 \\ \hline 2 & 2 & 7 & 9 \\ \hline 6 & 8 & 10 & 11 \\ \hline \end{array} \text{ and } \begin{array}{|c|}\hline 1 \\ \hline 3 \\ \hline 6 \\ \hline \end{array} \sim \begin{array}{|c|c|c|}\hline 1 & 2 & 3 \\ \hline 3 & 3 & 4 \\ \hline 4 & 5 & 6 \\ \hline \end{array}.$$

A one-column tableau is called a fundamental tableau if its content is $[i, i+k] \setminus \{r\}$ for $r \in \{i+1, \ldots, i+k-1\}$. A tableau T is said to have small gaps if each of its columns is a fundamental tableau. Any tableau in $\mathrm{SSYT}(k,[n])$ is $\sim$-equivalent to a unique small gap tableau.

2.3. Dominance order

Let $\lambda = (\lambda_1, \ldots, \lambda_\ell)$ with $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_\ell \geq 0$ and $\mu = (\mu_1, \ldots, \mu_\ell)$ with $\mu_1 \geq \mu_2 \geq \cdots \geq \mu_\ell \geq 0$ be partitions. Then $\lambda \geq \mu$ in dominance order if $\sum_{j \leq i} \lambda_j \geq \sum_{j \leq i} \mu_j$ for $i = 1, \ldots, \ell$. For a tableau T (, not necessarily rectangular shape), let $\mathrm{sh}(T)$ denote the shape of T. That is, $\mathrm{sh}(T) = (\lambda_1, \ldots, \lambda_r)$, where λ_i is the number of boxes of T in the ith row. For $i \in [n]$, let $T[i]$ denote the restriction of $T \in \mathrm{SSYT}(k,[n])$ to the entries in $[i]$. That is, $T[i]$ is the tableau obtained from T by removing all boxes which have numbers greater than i.

For a tableau T, we call the multi-set of numbers appearing (count multiplicities) in T the content of T. For $T, T' \in \mathrm{SSYT}(k,[n])$ with the same content, we say that $T \geq T'$ if $\mathrm{sh}(T[i]) \geq \mathrm{sh}(T'[i])$ in the dominance order on partitions, for $i = 1, \ldots, n$.

2.4. Cluster variables in $\mathbb{C}[\mathrm{Gr}(k,n)]$

Recall that $\mathbb{C}[\mathrm{Gr}(k,n,\sim)]$ is the quotient of $\mathbb{C}[\mathrm{Gr}(k,n)]$ by the ideal

$$\langle P_{i,i+1,\ldots,i+k-1} - 1, \quad i \in [n-k+1] \rangle.$$

Theorem 3.25 in [13] states that every cluster variable in the cluster algebra $\mathbb{C}[\mathrm{Gr}(k,n,\sim)]$ is of the form $\mathrm{ch}(T)$ (see Equation (1); the notation $\mathrm{ch}(T)$ is used because it corresponds to the q-character of a module of the quantum affine algebra $U_q(\widehat{\mathfrak{sl}_k})$) for some real prime $T \in \mathrm{SSYT}(k,[n])$ (a tableau is called real (resp. prime) if the corresponding quantum affine algebra module is real (resp. prime), [13]). An explicit formula of $\mathrm{ch}(T)$ is given in Theorem 5.8 of [13]:

$$\mathrm{ch}(T) = \sum_{u \in S_k} (-1)^{\ell(uw_T)} p_{uw_0, w_T w_0}(1) P_{u;T'} \in \mathbb{C}[\mathrm{Gr}(k,n,\sim)] \tag{1}$$

where T' is the small gap tableau such that $T \sim T'$, $P_{u;T'}$ is some monomial of Plücker coordinates, w_T is some permutation in S_k, and $p_{u,v}(q)$ is a Kazhdan-Lusztig polynomial. When $\mathrm{ch}(T)$ is a cluster variable, we also call T itself a cluster variable.

Remark 2.2. The set of cluster variables in $\mathbb{C}[\mathrm{Gr}(k,n)]$ is the union of the set of cluster variables in $\mathbb{C}[\mathrm{Gr}(k,n,\sim)]$ and frozen variables in $\mathbb{C}[\mathrm{Gr}(k,n)]$. The frozen variables (up to sign) in $\mathbb{C}[\mathrm{Gr}(k,n)]$ are $P_{i,i+1,\ldots,i+k-1}$, $i \in [n]$, where $i+n$ are identified with i. The frozen variables correspond to one-column tableaux with consecutive entries or with entries $\{1,2,\ldots,r,n-k+r+1,\ldots,n-1,n\}$, $r \in [k-1]$.

We compute cluster variables in $\mathbb{C}[\mathrm{Gr}(k,n)]$ in the following way [13, Section 4]: Starting from the initial seed of $\mathbb{C}[\mathrm{Gr}(k,n)]$, each time we perform a mutation at the cluster variable $\mathrm{ch}(T_r)$, we obtain a cluster variable $\mathrm{ch}(T'_r)$ defined recursively by

$$\mathrm{ch}(T'_r)\mathrm{ch}(T_r) = \prod_{i \to r} \mathrm{ch}(T_i) + \prod_{r \to i} \mathrm{ch}(T_i),$$

with $\mathrm{ch}(T_i)$ the cluster variable at the vertex i. Denote by $\max\{\cup_{i\to r} T_i, \cup_{r \to i} T_i\}$ the tableau which is larger in the dominance order. The tableau T' corresponding to the new cluster variable $\mathrm{ch}(T'_r)$ can be computed by the following formula:

$$T'_r = T_r^{-1} \max\{\cup_{i\to r} T_i, \cup_{r \to i} T_i\}. \tag{2}$$

The following are some examples of mutations in $\mathbb{C}[\mathrm{Gr}(3,8)]$:

$$\mathrm{ch}(\begin{array}{|c|}\hline 1\\\hline 3\\\hline 4\\\hline\end{array})\mathrm{ch}(\begin{array}{|c|}\hline 2\\\hline 3\\\hline 5\\\hline\end{array}) = \mathrm{ch}(\begin{array}{|c|}\hline 1\\\hline 3\\\hline 5\\\hline\end{array})\mathrm{ch}(\begin{array}{|c|}\hline 2\\\hline 3\\\hline 4\\\hline\end{array}) + \mathrm{ch}(\begin{array}{|c|}\hline 1\\\hline 2\\\hline 3\\\hline\end{array})\mathrm{ch}(\begin{array}{|c|}\hline 3\\\hline 4\\\hline 5\\\hline\end{array}),$$

$$\mathrm{ch}(\begin{array}{|c|}\hline 2\\\hline 3\\\hline 8\\\hline\end{array})\mathrm{ch}(\begin{array}{|c|c|c|}\hline 1&3&4\\\hline 2&5&6\\\hline 4&7&8\\\hline\end{array}) = \mathrm{ch}(\begin{array}{|c|}\hline 1\\\hline 2\\\hline 8\\\hline\end{array})\mathrm{ch}(\begin{array}{|c|c|}\hline 3&4\\\hline 5&6\\\hline 7&8\\\hline\end{array})\mathrm{ch}(\begin{array}{|c|}\hline 2\\\hline 3\\\hline 4\\\hline\end{array}) + \mathrm{ch}(\begin{array}{|c|}\hline 3\\\hline 4\\\hline 8\\\hline\end{array})\mathrm{ch}(\begin{array}{|c|c|}\hline 2&4\\\hline 5&6\\\hline 7&8\\\hline\end{array})\mathrm{ch}(\begin{array}{|c|}\hline 1\\\hline 2\\\hline 3\\\hline\end{array}).$$

3. Cluster variables in Grassmannian cluster algebras

In this section, we describe the result of our computations of cluster variables in $\mathbb{C}[\mathrm{Gr}(k,n)]$, giving rise to our dataset.

We generate cluster variables by performing random mutations and following formula (2). When we compute cluster variables with ranks less than or equal to a given number r, if we see a cluster variable (tableau) with rank greater than r, we mutate at that vertex again so that the cluster variable with rank greater than r does not appear. In this way, the cluster variables we generate are always with ranks less or equal to r. In practice, we perform sufficiently many mutations and when we see that no new cluster variables appear after ~ 10000 core hours on the HPC cluster (approximately 10% of the total core time per run), we conjecture that we have obtained all cluster variables with ranks less than or equal to r.

3.1. Some finite-type cluster algebras

The cluster algebra $\mathbb{C}[\mathrm{Gr}(3,3)]$ has only one frozen variable

$$\begin{array}{|c|}\hline 1\\\hline 2\\\hline 3\\\hline\end{array},$$

and no mutable cluster variables.

There are 4 frozen variables in $\mathbb{C}[\mathrm{Gr}(3,4)]$:

$$\begin{array}{|c|}\hline 1\\\hline 2\\\hline 3\\\hline\end{array},\ \begin{array}{|c|}\hline 1\\\hline 2\\\hline 4\\\hline\end{array},\ \begin{array}{|c|}\hline 1\\\hline 3\\\hline 4\\\hline\end{array},\ \begin{array}{|c|}\hline 2\\\hline 3\\\hline 4\\\hline\end{array}$$

and no mutable cluster variables.

In $\mathbb{C}[\mathrm{Gr}(3,5)]$, there are 5 cluster variables:

$$\begin{array}{|c|}\hline 1\\\hline 3\\\hline 5\\\hline\end{array},\ \begin{array}{|c|}\hline 2\\\hline 3\\\hline 5\\\hline\end{array},\ \begin{array}{|c|}\hline 2\\\hline 4\\\hline 5\\\hline\end{array},\ \begin{array}{|c|}\hline 1\\\hline 2\\\hline 4\\\hline\end{array},\ \begin{array}{|c|}\hline 1\\\hline 3\\\hline 4\\\hline\end{array},$$

and 5 frozen variables:

$$\begin{array}{|c|}\hline 1\\\hline 2\\\hline 3\\\hline\end{array},\ \begin{array}{|c|}\hline 2\\\hline 3\\\hline 4\\\hline\end{array},\ \begin{array}{|c|}\hline 3\\\hline 4\\\hline 5\\\hline\end{array},\ \begin{array}{|c|}\hline 1\\\hline 2\\\hline 5\\\hline\end{array},\ \begin{array}{|c|}\hline 1\\\hline 4\\\hline 5\\\hline\end{array}.$$

In $\mathbb{C}[\mathrm{Gr}(3,6)]$, there are 16 cluster variables:

$$\begin{array}{|c|}\hline 1\\\hline 4\\\hline 6\\\hline\end{array},\ \begin{array}{|c|}\hline 3\\\hline 4\\\hline 6\\\hline\end{array},\ \begin{array}{|c|}\hline 2\\\hline 4\\\hline 6\\\hline\end{array},\ \begin{array}{|c|}\hline 2\\\hline 3\\\hline 6\\\hline\end{array},\ \begin{array}{|c|}\hline 2\\\hline 5\\\hline 6\\\hline\end{array},\ \begin{array}{|c|}\hline 2\\\hline 4\\\hline 5\\\hline\end{array},\ \begin{array}{|c|}\hline 2\\\hline 3\\\hline 5\\\hline\end{array},\ \begin{array}{|c|}\hline 1\\\hline 2\\\hline 5\\\hline\end{array},\ \begin{array}{|c|}\hline 1\\\hline 4\\\hline 5\\\hline\end{array},\ \begin{array}{|c|}\hline 1\\\hline 3\\\hline 5\\\hline\end{array},\ \begin{array}{|c|}\hline 1\\\hline 3\\\hline 4\\\hline\end{array},\ \begin{array}{|c|}\hline 1\\\hline 3\\\hline 6\\\hline\end{array},\ \begin{array}{|c|}\hline 3\\\hline 5\\\hline 6\\\hline\end{array},\ \begin{array}{|c|}\hline 1\\\hline 2\\\hline 4\\\hline\end{array},\ \begin{array}{|c|c|}\hline 1&3\\\hline 2&5\\\hline 4&6\\\hline\end{array},\ \begin{array}{|c|c|}\hline 1&2\\\hline 3&4\\\hline 5&6\\\hline\end{array},$$

and 6 frozen variables:

$$\begin{array}{|c|}\hline 1\\\hline 2\\\hline 3\\\hline\end{array},\ \begin{array}{|c|}\hline 2\\\hline 3\\\hline 4\\\hline\end{array},\ \begin{array}{|c|}\hline 3\\\hline 4\\\hline 5\\\hline\end{array},\ \begin{array}{|c|}\hline 4\\\hline 5\\\hline 6\\\hline\end{array},\ \begin{array}{|c|}\hline 1\\\hline 5\\\hline 6\\\hline\end{array},\ \begin{array}{|c|}\hline 1\\\hline 2\\\hline 6\\\hline\end{array}.$$

In $\mathbb{C}[\mathrm{Gr}(3,7)]$, there are 28 one-column tableaux which are cluster variables and 7 one-column tableaux which are frozen variables. There are 14 rank 2 cluster variables which are obtained by sending $i \mapsto a_i$ in

$$\begin{array}{|c|c|}\hline 1&3\\\hline 2&5\\\hline 4&6\\\hline\end{array},\ \begin{array}{|c|c|}\hline 1&2\\\hline 3&4\\\hline 5&6\\\hline\end{array},$$

where $a_1 < \cdots < a_6 \in \{1, \ldots, 7\}$.

In $\mathbb{C}[\mathrm{Gr}(3,8)]$, there are 48 one-column tableaux which are cluster variables and 8 one-column tableaux which are frozen variables. There are 56 rank 2 cluster variables which are obtained by sending $i \mapsto a_i$ in the tableaux

$$\begin{array}{|c|c|}\hline 1&3\\\hline 2&5\\\hline 4&6\\\hline\end{array},\ \begin{array}{|c|c|}\hline 1&2\\\hline 3&4\\\hline 5&6\\\hline\end{array},$$

where $a_1 < \cdots < a_6 \in \{1, \ldots, 8\}$. There are 24 rank 3 cluster variables:

$$\begin{array}{|c|c|c|}\hline 1&3&4\\\hline 2&5&6\\\hline 4&7&8\\\hline\end{array}, \begin{array}{|c|c|c|}\hline 1&2&4\\\hline 2&3&7\\\hline 5&6&8\\\hline\end{array}, \begin{array}{|c|c|c|}\hline 1&2&3\\\hline 4&5&6\\\hline 6&7&8\\\hline\end{array}, \begin{array}{|c|c|c|}\hline 1&1&3\\\hline 2&5&6\\\hline 4&7&8\\\hline\end{array}, \begin{array}{|c|c|c|}\hline 1&3&4\\\hline 2&6&7\\\hline 5&7&8\\\hline\end{array}, \begin{array}{|c|c|c|}\hline 1&2&3\\\hline 3&4&5\\\hline 6&7&8\\\hline\end{array}, \begin{array}{|c|c|c|}\hline 1&2&5\\\hline 3&4&7\\\hline 6&8&8\\\hline\end{array}, \begin{array}{|c|c|c|}\hline 1&2&5\\\hline 3&4&7\\\hline 5&6&8\\\hline\end{array},$$

$$\begin{array}{|c|c|c|}\hline 1&2&3\\\hline 4&4&5\\\hline 6&7&8\\\hline\end{array}, \begin{array}{|c|c|c|}\hline 1&3&4\\\hline 2&6&7\\\hline 5&8&8\\\hline\end{array}, \begin{array}{|c|c|c|}\hline 1&3&4\\\hline 2&5&6\\\hline 5&7&8\\\hline\end{array}, \begin{array}{|c|c|c|}\hline 1&2&5\\\hline 3&4&7\\\hline 6&6&8\\\hline\end{array}, \begin{array}{|c|c|c|}\hline 1&2&3\\\hline 2&5&6\\\hline 4&7&8\\\hline\end{array}, \begin{array}{|c|c|c|}\hline 1&2&3\\\hline 4&5&6\\\hline 7&7&8\\\hline\end{array}, \begin{array}{|c|c|c|}\hline 1&1&2\\\hline 3&4&5\\\hline 6&7&8\\\hline\end{array}, \begin{array}{|c|c|c|}\hline 1&2&4\\\hline 3&3&7\\\hline 5&6&8\\\hline\end{array},$$

$$\begin{array}{|c|c|c|}\hline 1&2&4\\\hline 3&4&7\\\hline 5&6&8\\\hline\end{array}, \begin{array}{|c|c|c|}\hline 1&2&5\\\hline 3&4&7\\\hline 6&7&8\\\hline\end{array}, \begin{array}{|c|c|c|}\hline 1&2&3\\\hline 4&5&5\\\hline 6&7&8\\\hline\end{array}, \begin{array}{|c|c|c|}\hline 1&2&2\\\hline 3&4&5\\\hline 6&7&8\\\hline\end{array}, \begin{array}{|c|c|c|}\hline 1&3&4\\\hline 2&6&6\\\hline 5&7&8\\\hline\end{array}, \begin{array}{|c|c|c|}\hline 1&2&3\\\hline 4&5&6\\\hline 7&8&8\\\hline\end{array}, \begin{array}{|c|c|c|}\hline 1&1&4\\\hline 2&3&7\\\hline 5&6&8\\\hline\end{array}, \begin{array}{|c|c|c|}\hline 1&3&3\\\hline 2&5&6\\\hline 4&7&8\\\hline\end{array}.$$

For general $k \leq n$, there are infinitely many cluster variables in $\mathbb{C}[\mathrm{Gr}(k,n)]$. On the other hand, there are finitely many cluster variables in $\mathbb{C}[\mathrm{Gr}(k,n)]$ with a fixed rank. For example, there are 168 rank 2 cluster variables and 225 rank 3 cluster variables in $\mathbb{C}[\mathrm{Gr}(3,9)]$, [9].

As a core part of the work in this project, we have computed databases which contain all cluster variables with certain ranks in $\mathbb{C}[\mathrm{Gr}(k,n)]$ for selected k, n (noting these naturally contain all lower ns). Specifically, we compute the cluster variables as semistandard Young tableaux in the Grassmannian cluster algebras $\mathbb{C}[\mathrm{Gr}(3,12)]$ up to rank 6, $\mathbb{C}[\mathrm{Gr}(4,12)]$ up to rank 4, and $\mathbb{C}[\mathrm{Gr}(4,10)]$ up to rank 6, totaling 2656212, 3089105, and 6346878 tableaux respectively. All datasets are available on GitHub.

3.2. Numbers of cluster variables with given ranks

We denote by $N_{k,n,r}$, the number of cluster variables (including frozen variables) of rank r in $\mathbb{C}[\mathrm{Gr}(k,n)]$. Using high-performance computing, we stochastically compute all cluster variables with certain rank in $\mathbb{C}[\mathrm{Gr}(k,n)]$. Since the process is stochastic we cannot explicitly verify this is an exhaustive list, however we note that in each case the last ~10% of runs did not generate any new variables. Therefore, the numbers $N_{k,n,r}$ in Table 1 provide at the very least lower bounds on the true number of cluster variables, and likely, equality.

It was proved in [9], that the number of rank 2 cluster variables in $\mathbb{C}[\mathrm{Gr}(k,n)]$ ($k \leq n/2$) is at least

$$N_{k,n,2} = \sum_{r=3}^{k} \left(\frac{2r}{3} \cdot p_1(r) + 2r \cdot p_2(r) + 4r \cdot p_3(r) \right) \cdot \binom{n}{2r} \binom{n-2r}{k-r}, \tag{3}$$

r	1	2	3	4	5	6	7	8	9	10
$N_{3,3,r}$	1	0	0	0	0	0	0	0	0	0
$N_{3,4,r}$	4	0	0	0	0	0	0	0	0	0
$N_{3,5,r}$	10	0	0	0	0	0	0	0	0	0
$N_{3,6,r}$	20	2	0	0	0	0	0	0	0	0
$N_{3,7,r}$	35	14	0	0	0	0	0	0	0	0
$N_{3,8,r}$	56	56	24	0	0	0	0	0	0	0
$N_{3,9,r}$	84	168	225	288	372	414	522	594	612	744
$N_{3,10,r}$	120	420	1170	3280	8200	19140				
$N_{3,11,r}$	165	924	4455	20504	77957	256553				
$N_{3,12,r}$	220	1848	13860	92980	486172	2061132				
$N_{4,4,r}$	1	0	0	0	0	0	0	0	0	0
$N_{4,5,r}$	5	0	0	0	0	0	0	0	0	0
$N_{4,6,r}$	15	0	0	0	0	0	0	0	0	0
$N_{4,7,r}$	35	14	0	0	0	0	0	0	0	0
$N_{4,8,r}$	70	120	174	208	296	304	420	416	536	480
$N_{4,9,r}$	126	576	2421	8622	27054	69390				
$N_{4,10,r}$	210	2040	17665	117930	597500	2353760				
$N_{4,11,r}$	330	5940	90563	980100						
$N_{4,12,r}$	495	15048	367479	5963856						

Table 1. Number of cluster variables in $\mathbb{C}[\mathrm{Gr}(k,n)]$ of rank r. Note each $N_{n,k,r}$ contains all those in $N_{n,k-1,r}$ by definition, so there are $(N_{n,k,r} - N_{n,k-1,r})$ new SSYT cluster variables for each box. Empty box entries denote variables to be computed in future work, and were beyond reasonable means for current computation.

where $p_i(r)$ is the number of partitions $r = r_1 + r_2 + r_3$ such that $r_1, r_2, r_3 \in \mathbb{Z}_{\geq 1}$ and $|\{r_1, r_2, r_3\}| = i$. According to our computation results, we expect that the number of rank 2 cluster variables in $\mathbb{C}[\mathrm{Gr}(k,n)]$ ($k \leq n/2$) is exactly given by formula (3).

According to our computation results, we also have the following conjectures:

Conjecture 3.1. *The corresponding number of cluster variables is given by the following expressions*

$$\begin{aligned}
N_{3,n,3} &= 24\binom{n}{8} + 9\binom{n}{9}, \\
N_{3,n,4} &= 288\binom{n}{9} + 400\binom{n}{10} + 264\binom{n}{11} + 48\binom{n}{12}, \\
N_{4,n,3} &= 174\binom{n}{8} + 855\binom{n}{9} + 1285\binom{n}{10} + 693\binom{n}{11} + 123\binom{n}{12}.
\end{aligned}$$

Conjecture 3.2. *For any tableau $T \in \mathrm{SSYT}(k,[n])$ with entries $a_1 < \ldots < a_r$ and any function $f : \{a_1, \ldots, a_r\} \to [n']$, $n' \geq n$, such that $f(a_1) < \ldots < f(a_r)$, we have that T is a cluster variable in $\mathbb{C}[\mathrm{Gr}(k,n)]$ if and only if $f(T)$ is a cluster variable in $\mathbb{C}[\mathrm{Gr}(k,n')]$.*

4. Machine Learning

With the increase in computational power over the preceding decades, the ability to generate large amounts of mathematical data has become far more manageable. Where past mathematical work has focused on conjecture formulation from computation by hand on smaller samples of selected examples, now with larger datasets and algorithms to perform analysis computationally, data science is steadily becoming a key player in mathematical research.

Machine learning (ML), an umbrella field encompassing a large range of techniques from supervised, unsupervised, and reinforcement learning, has already seen a large amount of success in mathematics and related areas [2, 6–8, 10–12, 16, 17, 26, 27, 31, 32, 34, 36, 39]. Of particular relevance here, is past work on the use of ML to examine exchange graphs describing cluster seed interrelations [5, 18], built on [37].

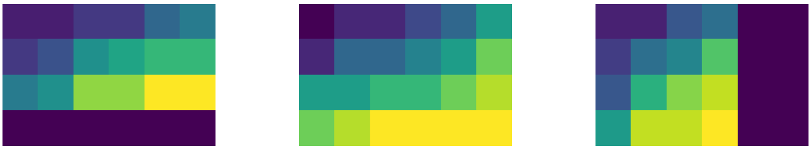
With the 3 large datasets of cluster variables represented as SSYT, specifically $\{\mathbb{C}[\mathrm{Gr}(3,12)]\ r6, \mathbb{C}[\mathrm{Gr}(4,10)]\ r6, \mathbb{C}[\mathrm{Gr}(4,12)]\ r4\}$ where $r\#$ denotes the maximum rank (number of tableau columns) considered, we now apply a variety of techniques from ML to analyse them.

4.1. Data Formatting

To ensure consistent formatting for data processing, all the SSYT were formatted as `numpy` arrays in `python`, padded with zeros up to the maximum size, such that they all had shape (4,6).

Examples of these arrays represented as images are shown in Figure 1, where the clear padding of the $k = 3$ cases (bottom row all zeros) and $r = 4$ cases (right two columns all zeros) are shown for the $\mathbb{C}[\mathrm{Gr}(3,12)]$ $r6$ and $\mathbb{C}[\mathrm{Gr}(4,12)]$ $r4$ datasets respectively. The lighter box in the $\mathbb{C}[\mathrm{Gr}(4,12)]$ $r4$ example indicates the higher maximum entry than $\mathbb{C}[\mathrm{Gr}(4,10)]$ $r6$ as $n = 12 > 10$.

These images just represent single examples from these Grassmannian cluster algebras. In each case lower rank SSYT are included (with more columns padded), as well as those with a smaller range of entries (where the colours are darker throughout the image).



(a) $\mathbb{C}[\mathrm{Gr}(3,12)]$, $r \leq 6$ (b) $\mathbb{C}[\mathrm{Gr}(4,10)]$, $r \leq 6$ (c) $\mathbb{C}[\mathrm{Gr}(4,12)]$, $r \leq 4$

Figure 1. Example images produced from the padded versions of the SSYT representing cluster variables in the respective Grassmannians. Note that for $\mathbb{C}[\mathrm{Gr}(k,n)]$ k represents the number of rows, n the maximum entry, and r the rank and hence the number of columns. These example images have the maximum rank in each case.

4.1.1. NCV Data Generation. A point of key importance is that not all SSYT represent cluster variables in Grassmannian cluster algebras. Therefore one can create tableaux that increase along the k rows, strictly increase down the r columns, and with maximum entry n, which do not correspond to a cluster variable in the Grassmannian $\mathbb{C}[\mathrm{Gr}(k,n)]$ with rank up to r.

Therefore, we refer back to Problem 1.1: can ML techniques discover a relation that allows them to identify this? However, prior to applying ML methods, non-cluster-variable data, which we denote 'NCV', corresponding to SSYT which are not cluster variables must be generated. Conversely, we denote the dataset of SSYT which are cluster variables as 'CV'.

For each Grassmannian CV dataset an equivalent NCV dataset was generated, such that the number of rows, maximum entry, and maximum number of columns matched the Grassmannian's k, n, r respectively. Before generating each tableau, the number of columns was sampled from $[1, r]$. The NCV SSYTs were then initialised as random arrays of entries in the range $[1, n]$, which were sorted (enforcing an increase along the rows), then columns were checked and entries regenerated until the condition of strictly increasing down columns was met. Exhaustive checks were then applied to ensure each NCV SSYT was: 1) not in the respective Grassmannian CV dataset; 2) not already generated. These exhaustive checks ensure that the data is truly an NCV SSYT. We generate 10,000 tableaux for each NCV dataset, to be compared with a random sample of 10,000 tableaux from each respective Grassmannian.

4.2. Supervised Classification

In this subsection the content of Problem 1.1 is addressed, namely: how well can ML architectures learn to distinguish SSYT from different Grassmannian cluster algebras, or SSYT which are cluster variables from those which are not?

The ML architectures considered in this work are Support Vector Machines (SVM) and dense feed-forward Neural Networks (NN).

The goal of SVMs is to find $p-1$-dimensional hypersurfaces that best separate data points of different classes in $\mathbb{R}^p$, where in our case $p = 24$ for the 24 entries of the SSYT. The shape of the hypersurface is dictated by the kernel style, and a regularisation parameter adds a cost to each parameter used to define the hypersurface – discouraging them from overfitting and becoming too complicated. The hypersurfaces are fitted using training data and later performance-tested with test data. The SVM we use here has regularisation parameter of 1.0 uses a Gaussian 'rbf' kernel. We train until we reach a tolerance of 0.001 for fractional improvement in the proportion of correct classifications.

NNs are designed for complex non-linear function fitting. They are built out of perceptrons which take a vector as input, then output a number via linear action followed by non-linear activation: output $= \text{act}\big(\sum_i (w_i \cdot \text{input}_i) + b\big)$ for weights w_i and bias b. Layers of these perceptrons all connected to the subsequent layer make the NN dense, and feed-forward, as data flows through the network from initial input, through the layers, to final output. The optimiser algorithm updates the weights and biases (by amounts proportional to the learning rate) during training to minimise the loss function, which is a measure of the difference between the NN predicted and the real output for each specific input, over batches of input data. The NN architecture we use consists of three layers of size 16, 32 and 16, the perceptrons in each layer use ReLU activation, and the network is trained to minimise log-loss using the Adam optimisation algorithm, with batch size 200 and a learning rate of 0.001 until convergence below a tolerance of 0.0001.

Both these architectures take the `sklearn` default hyperparameter values [38].

The learning performance is measured using accuracy and Matthew's correlation coefficient (MCC) on the test set predictions. These metrics may be described as functions on the confusion matrix (CM). Accuracy is the proportion of predictions that are correctly classified (i.e., the normalised sum of the diagonal of the confusion matrix). MCC is an analogue of this that accounts for off-diagonal terms, such that dataset bias is avoided in altering the validity of the measure. The learning is carried out using 5-fold cross-validation, meaning that we train and test the network on 5 different partitions of the data, such that the union of the test sets equals the full dataset. This produces a set of 5 results for each learning measure, from which we compute an average and the standard error.

The first investigation uses NNs to perform multiclassification between the SSYT of the 3 Grassmannian CV datasets, whilst the second uses both SVMs and NNs to perform binary classification between the CV and NCV SSYT for each dataset. Due to the computational demands of training, random samples of 10,000 tableaux were taken from each Grassmannian CV dataset, to match the sizes of the generated NCV datasets.

4.2.1. Grassmannian Multiclassification. Before performing the NN supervised multiclassification between the 3 Grassmannian CV databases, it is noted that the $\mathbb{C}[\mathrm{Gr}(4,10)]$ $r6$ and $\mathbb{C}[\mathrm{Gr}(4,12)]$ $r4$ databases have natural overlap of tableaux in $\mathbb{C}[\mathrm{Gr}(4,10)]$ $r4$. Using Table 1, the two Grassmannians have 137845 variables in common; hence $137845/3089105 \sim 0.04$ of the $\mathbb{C}[\mathrm{Gr}(4,10)]$ $r6$ data is in $\mathbb{C}[\mathrm{Gr}(4,12)]$ $r4$ and $137845/6346878 \sim 0.02$ of the $\mathbb{C}[\mathrm{Gr}(4,12)]$ $r4$ data is in $\mathbb{C}[\mathrm{Gr}(4,10)]$ $r6$. Therefore to avoid multi-labelling of tableaux for this classification problem the $\mathbb{C}[\mathrm{Gr}(4,10)]$ $r4$ tableaux were removed from both relevant datasets.

The learning measures, with standard error over the cross-validation, to 3 decimal places were:

$$Accuracy = 1.000 \pm 0.000\ , \tag{4}$$

$$MCC = 1.000 \pm 0.000\ , \tag{5}$$

$$CM = \begin{pmatrix} 0.333 \pm 0.000 & 0.000 \pm 0.000 & 0.000 \pm 0.000 \\ 0.000 \pm 0.000 & 0.333 \pm 0.000 & 0.000 \pm 0.000 \\ 0.000 \pm 0.000 & 0.000 \pm 0.000 & 0.333 \pm 0.000 \end{pmatrix}\ , \tag{6}$$

where in the confusion matrix entries CM_{ij} have i index as the true class and j index as the predicted class. The three Grassmannian classes are $(1,2,3) = (\mathbb{C}[\mathrm{Gr}(3,12)]r6, \mathbb{C}[\mathrm{Gr}(4,10)]r6, \mathbb{C}[\mathrm{Gr}(4,12)]r4)$ respectively.

These results show perfect performance in identifying the Grassmannian a tableau belongs to. This is reassuring behaviour as already by eye one can distinguish the $\mathbb{C}[\mathrm{Gr}(3,12)]$ tableaux by the number of rows, as well as the tableaux from the two $k = 4$ databases due to the number of columns (i.e. rank) or maximum entry.

4.2.2. Binary Classification of Cluster Variables from SSYT. Learning measures for both SVM and NN architectures performing binary classification between the Grassmannian CV data and respective NCV data are shown in Table 2.

Architecture	Learning Measure	Grassmannian		
		$\mathbb{C}$[Gr(3,12)] $r6$	$\mathbb{C}$[Gr(4,10)] $r6$	$\mathbb{C}$[Gr(4,12)] $r4$
SVM	Accuracy	0.913 ± 0.002	0.928 ± 0.001	0.925 ± 0.001
	MCC	0.830 ± 0.004	0.867 ± 0.004	0.852 ± 0.002
NN	Accuracy	0.938 ± 0.002	0.946 ± 0.002	0.941 ± 0.002
	MCC	0.878 ± 0.003	0.893 ± 0.005	0.885 ± 0.004

Table 2. Supervised binary classification between CV SSYT representing cluster variables in the respective Grassmannians, and NCV generated matrices designed to mimic them.

Both architecture styles are incredibly successful at determining the cluster variables from the full sets of SSYT. However, the still exceptional performance of the SVMs indicates that there is likely some unknown implicit structure in the SSYT entries that make them cluster variables.

In each case the NN architecture performs better, as may be expected since the architecture is more general. The MCC scores correlate with accuracy, which is reassuring that the data is representative and unbiased, whilst the better performance for $\mathbb{C}[\mathrm{Gr}(4,10)]$ $r6$ over $\mathbb{C}[\mathrm{Gr}(4,12)]$ $r4$ implies that rank is a more important feature for determining the cluster variable property. Explicit analysis of the misclassified SSYT in each case show no discernible pattern in the tableaux, confirming that the architectures are picking up on a more subtle structure for their learning.

These results confidently answer Problem 1.1 affirmatively: ML *can* pick up on the underlying structure that makes a SSYT a cluster variable.

4.3. Principal Component Analysis

While supervised learning methods are better adapted to address the classification-style of Problem 1.1, techniques from the ML subfield of unsupervised learning are better suited to extracting such underlying structure in the data as desired for Problem 1.2.

The first of the techniques we consider is Principal Component Analysis (PCA). As a technique, PCA extracts the most important features of a dataset through diagonalisation of the covariance matrix between the data dimensions across the dataset. Identifying the eigenvectors of the covariance matrix and sorting them by decreasing eigenvalue, the data points can be projected onto their most significant principal components which best describe the most variance – and hence structure – in the data. These eigenvalues can also be reinterpreted as the explained variance, giving the amount of variance explained by each component. It is often useful to consider these explained variances as ratios, such that they have been normalised to sum to 1.

Traditional PCA, as just described, may be generalised to kernel PCA. There, the data points are conceptually mapped to a higher-dimensional space where distinguishing them becomes substantially easier due to the larger number of degrees of freedom. Then the principal components in this space can be computed, the transformed data points projected onto them, and mapped back to the original space. However, these higher-dimensional computations are costly and can in fact be avoided altogether by using the 'kernel trick'. The trick combines the above steps by defining a kernel that represents this mapping and projection, circumventing the need to actually compute in the higher-dimensional space in practice.

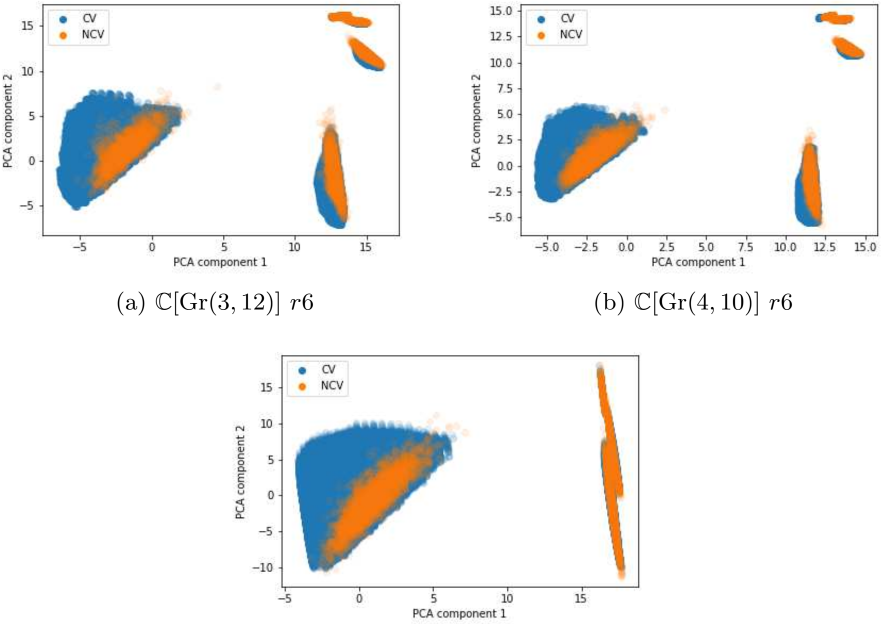
Whereas traditional PCA acts effectively with a linear mapping and hence linear kernel, kernel PCA can introduce non-linearity into the principal components, and hence identify non-linear structure in the data.

As a testing ground, we perform PCA (i.e. linear kernel PCA) on the Grassmannian CV data and equivalent NCV datasets. In each case the 24 explained variances over this 24-dimensional data have largest 2 normalised values

$$\begin{aligned}
\mathbb{C}[\mathrm{Gr}(3,12)]\ r6 &\implies (0.592, 0.138)\ ,\\
\mathbb{C}[\mathrm{Gr}(4,10)]\ r6 &\implies (0.631, 0.139)\ ,\\
\mathbb{C}[\mathrm{Gr}(4,12)]\ r4 &\implies (0.488, 0.179)\ ,
\end{aligned}$$

respectively. These clearly dominate the data structure (other explained variance ratios are all at least an order of magnitude smaller), and are respectively plotted as 2-dimensional plots in Figure 2.

As shown in each of these plots the NCV data (10,000 tableaux) sits nicely within the projections of the much larger Grassmannian CV datasets. This again emphasises that the NCV data is representative in the principal



(a) $\mathbb{C}[\mathrm{Gr}(3,12)]$ $r6$

(b) $\mathbb{C}[\mathrm{Gr}(4,10)]$ $r6$

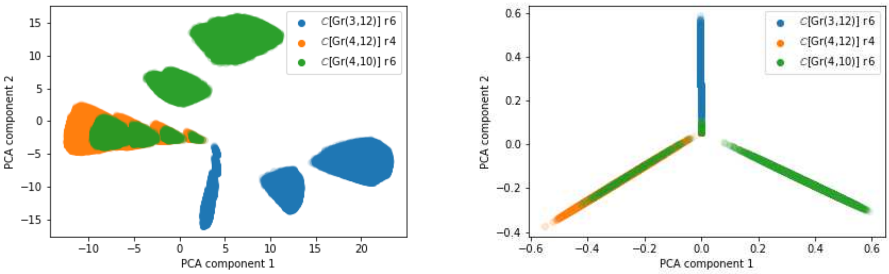
(c) $\mathbb{C}[\mathrm{Gr}(4,12)]$ $r4$

Figure 2. PCA decomposition (linear kernel) of the SSYT CV Grassmannian and NCV data for each of the respective datasets. The PCA shows that the NCV data generation is representative in the principal components.

components, and hence there is no significant linear structure that the supervised architectures can take advantage of in order to learn to distinguish the NCV data. It also further supports the point that the property that distinguishes cluster from NCV SSYT is more subtle, and hence it is even more impressive that the ML methods can pick up on it so successfully.

The clear clustering of each dataset is intriguing behaviour in itself – one we will further analyse in the following subsections.

4.3.1. PCA Clustering. Performing PCA on all the Grassmannian CV datasets together produces an amalgamation of the aforeseen individual PCA plots for each dataset. This PCA had dominant two explained variance ratios (0.592, 0.214), again reinforcing the 2D plotting of the two most significant principal components. These two components are shown in Figure 3a, and the equivalent two for a Gaussian 'rbf' kernel in Figure 3b.



(a) Linear kernel

(b) Gaussian kernel

Figure 3. PCA decomposition of the SSYT data for the 3 Grassmannians, using (a) linear and (b) Gaussian kernels respectively. Note there is significant overlap between $\mathbb{C}[\mathrm{Gr}(4,10)]$ $r6$ and $\mathbb{C}[\mathrm{Gr}(4,12)]$ $r4$ as expected, and cluster separation is largely due to padding – hence correctly clustering according to rank. The Gaussian kernel PCA was computed over a sample of 10,000 CV SSYT from each Grassmannian due to memory limits with the full datasets.

The Gaussian kernel PCA was performed over samples of 10,000 SSYT per dataset to allow feasible computation.

The linear PCA shows a clear separation of the $\mathbb{C}[\mathrm{Gr}(3,12)]$ $r6$ data, and a majority of the $\mathbb{C}[\mathrm{Gr}(4,10)]$ $r6$ data separating from the $\mathbb{C}[\mathrm{Gr}(4,12)]$ $r4$ data. This behaviour indicates simple structure differentiating the SSYT in each Grassmannian, which we may expect since the padding of the bottom row of $\mathbb{C}[\mathrm{Gr}(3,12)]$ $r6$ data clearly separates it, whilst equally padding of the rightmost two columns for the majority of the $\mathbb{C}[\mathrm{Gr}(4,12)]$ $r4$ data helps identify that data (where the maximum entry > 10). There is overlap in the leftmost part of the linear PCA plot, where the two $k = 4$ Grassmannians have common data. This was computationally confirmed to be exactly and exclusively the overlap data $\mathbb{C}[\mathrm{Gr}(4,10)]$ $r4$. This separation supports the exceptional results in §4.2.1 where the Grassmannians could be easily distinguished.

We delay the analysis of the separation and shapes of the clusters within each Grassmannian until the next subsection. Having tried other kernels for all the Grassmannian data, the pattern appeared most striking for a Gaussian kernel. In this kernel PCA the Grassmannians are clearly separated into symmetrically distributed lines, with some overlap of the low n and low rank tableaux where $k = 4$, and additionally some surprising overlap with the $\mathbb{C}[\mathrm{Gr}(3,12)]$ $r6$ data. Since the kernel was Gaussian this indicates the patches in Figure 3a were approximately Gaussian distributed, which reduce to a linear parameter in each case for each dataset's line.

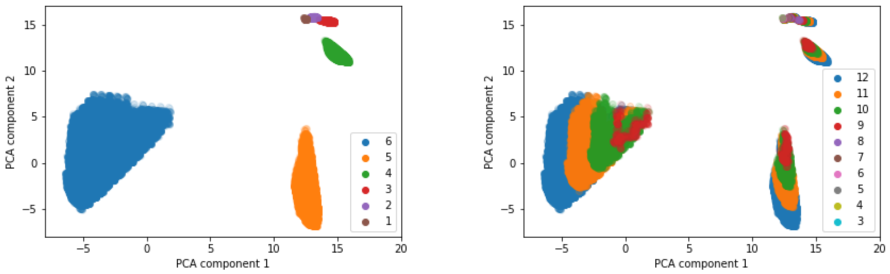
4.3.2. Dissecting the Clusters. The clear separation of the individual clusters for each Grassmannian CV dataset emblematises significant data structure. The cause of this clustering should be simple, and is well shown by the linear PCA plots for the rank-partitioning of the $\mathbb{C}[\mathrm{Gr}(3,12)]$ data in Figure 4a: namely, each cluster corresponds to a different rank of the data, simply identifiable by the padding, and hence easily leading to cluster separation.

The cluster relative sizes and shapes are more interesting, and are manifestly represented by the n-partitioning of the data in Figure 4b, which gives the clusters a mussel-like appearance. These plots show that all the clusters exhibit higher $n \geq 9$ tableaux; however, only the smallest cluster has data for $n \leq 8$. This is likely due to the fact that as one goes to larger ranks there are more tableau boxes to fill which require a higher maximum number to satisfy the SSYT conditions. This split can be attributed to the fact that many more ranks can be used to construct tableaux when $n > 8$, as shown in Table 1. This table also shows why no $n \leq 8$ tableaux appear in the larger clusters, as these exclusively correspond to higher rank data.

The cluster sizes correlate with the rank; this may be expected since higher rank tableaux have more entries and thus more combinations of numbers are available. However a priori, since we know that not all SSYT are cluster variables one may not expect – although we can build more SSYT at higher rank – that there would also be more cluster variables; this analysis shows that this is the case.

The cluster shapes show large amounts of overlap between n-partitions in the data, where each tableau appears to have a counterpart with a higher n; one may imagine this to be due to each tableau with maximum entry (say 9) being mapped on top of an identical tableau with all the same entries except that the final largest box has entry 10, 11, or 12. As the n value increases by one there is also a large number of new tableaux that can be created by increasing the largest entry to this new value, then trialing all combinations of increasing preceding entries by 1. This is what causes the clusters to grow as n is increased.

This clustering behaviour is repeated in the other two datasets also, with near-identical appearance, indicative of the structure being relevant to all Grassmannian CV data, and not specifically the $\mathbb{C}[\mathrm{Gr}(3,12)]$ data shown in Figure 4.



(a) $\mathbb{C}[\mathrm{Gr}(3,12)]$ partitioned according to $r \in [1,6]$

(b) $\mathbb{C}[\mathrm{Gr}(3,n)]$ $r6$ partitioned according to $n \in [3,12]$

Figure 4. PCA decomposition (linear kernel) of the $\mathbb{C}[\mathrm{Gr}(3,12)]$ SSYT data, plotted with partitions according to the rank r or maximum entry n. The PCA shows that the clusters separate according to rank, whilst the differing values of n expand the cluster sizes, akin to a mussel. Equivalent behaviour also holds for the other Grassmannians considered.

Therefore we can conclude that linear PCA can be used to distinguish clusters of Grassmannian cluster variables as SSYT according to their rank and k values, and gives an indication of the correlations with n. However, it cannot distinguish the NCV SSYT data from the CV SSYT data.

4.4. K-Means Clustering

An alternative unsupervised ML technique used for clustering is K-Means. This takes initialised centres for a preset number of clusters and aims to minimise the squared distance between each data point and its nearest cluster (whose sum is the inertia $\mathcal{I}$). It does this by iteratively allocating all data points to their nearest cluster, then replacing that cluster centre with the centroid of the cluster, then reallocating closest clusters for each data point.

Clustering performance is measured with inertia $\mathcal{I}$, which is the total Euclidean squared distance between each point and its closest cluster centre. Further to the full sum, inertia may also be normalised in various ways to improve interpretability. The two normalisation methods considered were: (1) divide by the total number of points and dimensions to give the average squared distance that a datapoint was from its closest cluster centre in a single dimension $\hat{\mathcal{I}}$; and (2) divide by total number of points and dimensions, and the range of data entries $\hat{\mathcal{I}}'$ to give a more relative version of (1).

Cluster	Grassmannian		
	$\mathbb{C}[\mathrm{Gr}(3,12)]$ $r6$	$\mathbb{C}[\mathrm{Gr}(4,10)]$ $r6$	$\mathbb{C}[\mathrm{Gr}(4,12)]$ $r4$
1	2656042	0	0
2	0	2951260	0
3	170	137845	6346878

Table 3. The distributions of the three Grassmannian CV datasets between the 3 clusters generated through the K-Means process.

The K-Means clustering algorithm used the `sci-kit learn` standard hyperparameters [38], such that 10 random initialisations are run to a convergence of 0.0001 tolerance in the inertia update or for a maximum of 300 iteration steps, with the best initialisation run selected.

To determine the optimal number of clusters to use, an elbow method is applied which reruns the clustering algorithm for a range of numbers of clusters and plots the inertia relative to the final inertia using just one cluster, and adding 0.01 × the number of clusters (so as to penalise using too many clusters). Whereas in traditional elbow methods the optimum number of clusters is represented by the largest change in gradient, in this formulation with the linear term in number of clusters added to the inertia, the lowest value across this range gives the optimal number of clusters.

4.4.1. Distinguishing Grassmannians. Whereas the PCA shows that the Grassmannian CV datasets can be well distinguished with simple linear structure using only a few components, we now investigate the use of K-Means on the full 24-dimensional tableaux vectors to further probe this observed clustering in the full-dimensional space. To first exemplify the utility of K-Means, we perform clustering for a concatenated list of all the tableaux across the three datasets, with a preset number of clusters of 3.

The K-Means algorithm converges, giving inertia measures:

$$\mathcal{I} = 672000000, \quad \hat{\mathcal{I}} = 2.32, \quad \hat{\mathcal{I}}' = 0.193, \tag{7}$$

to three significant figures. Dissecting how each of the datasets split between the clusters leads to the distributions shown in Table 3.

The inertia results are best interpreted using $\hat{\mathcal{I}}'$, where after clustering has converged, each datapoint is on average $< 20\%$ of the range of tableaux entries away from its closest cluster centre in each dimension. Since it has been established that there is already a noticeable overlap between the datasets this clustering is quite strong, and exemplifies the power of this technique in high-dimensional clustering.

The distributions of the cluster allocations for tableaux in each dataset, as shown in Table 3, solidify the algorithm's ability to distinguish tableaux according to the Grassmannian they relate to. The misclassifications where tableaux in the same Grassmannian were put in a different cluster happened with proportions 0.00006, 0.045, 0 for each of the $\mathbb{C}[\mathrm{Gr}(3,12)]$ $r6$, $\mathbb{C}[\mathrm{Gr}(4,10)]$ $r6$, $\mathbb{C}[\mathrm{Gr}(4,12)]$ $r4$ datasets respectively. Further explicit analysis shows those misclassified for $\mathbb{C}[\mathrm{Gr}(3,12)]$ $r6$ all had rank ≤ 2 and likely the large number of zeros for these padded tableaux threw off the clustering, whilst those misclassified for $\mathbb{C}[\mathrm{Gr}(4,10)]$ $r6$ all had rank ≤ 4 and were *exactly* the 137845 tableaux in the overlap with $\mathbb{C}[\mathrm{Gr}(4,12)]$ $r4$.

4.4.2. Distinguishing Cluster Tableaux. Now using the K-Means clustering method to probe the structure differentiating SSYT which are cluster variables from those which are not, each Grassmannian CV dataset is compared against its respective NCV dataset.

Manually setting 2 clusters did not partition as well the full list of all SSYT (both cluster variables, and non-cluster variables) into their respective classes for each of the Grassmannians. Although relatively more weight was put into one of the clusters for each case. These partitions are shown in Table 4, and show that $\sim 20\%$ of the data is misclassified under the clustering in each case. Explicit analysis for each of the Grassmannians (where r_{max} indicates the maximum rank in the dataset) shows that the misclassified NCV tableaux were all of rank r_{max}, which were actually *all* of the NCV tableaux with rank r_{max} in the NCV datasets, whilst the misclassified CV tableaux were all rank $< r_{max}$, being *all* the rank $< r_{max}$ tableaux in the respective datasets. Therefore although Table 4 appears to show clustering performance related to this property, it is only an artefact of the clustering algorithm partitioning off the largest rank.

To further investigate this K-Means on the clustering behaviour the elbow method was applied to identify the optimum number of clusters for partitioning the CV from NCV tableaux for the $\mathbb{C}[\mathrm{Gr}(3,12)]$ $r6$ dataset. The scaled inertia (relative to inertia with 1 cluster) is plotted for varying numbers of clusters in Figure 5.

The optimum produced by this process was 9 clusters, although as can be seen from the graph there is no obvious optimum as the performance plateaus such that adding additional clusters does not improve the clustering performance. These results clearly show that the K-Means algorithm cannot find structure in these datasets that leads to an obvious clustering,

Cluster	Grassmannian					
	$\mathbb{C}[\mathrm{Gr}(3,12)]$ $r6$		$\mathbb{C}[\mathrm{Gr}(4,10)]$ $r6$		$\mathbb{C}[\mathrm{Gr}(4,12)]$ $r4$	
	CV	NCV	CV	NCV	CV	NCV
1	2061132	595080	2352760	735345	5963856	383022
2	1969	8031	2311	7689	3254	6746

Table 4. The distributions of the 'CV' cluster variable SSYT and the 'NCV' non-cluster variable SSYT between the 2 clusters generated through the K-Means process, for each of the Grassmannian datasets respectively.

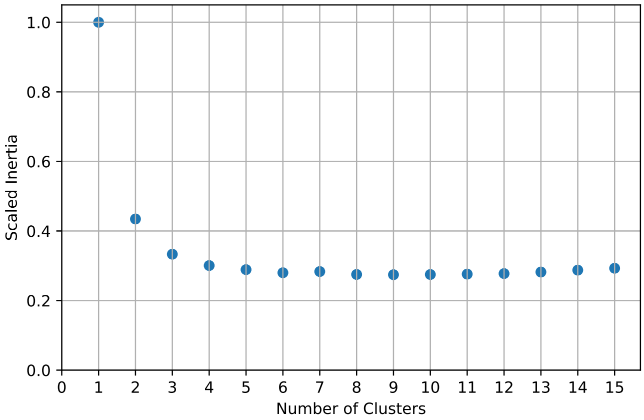


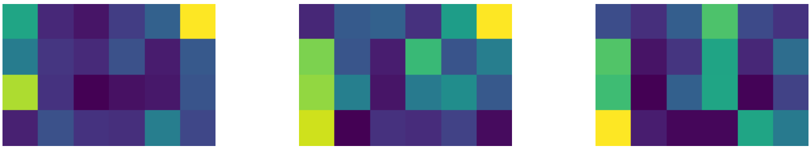
Figure 5. The elbow method for determining the optimum number of K-Means clusters when clustering the $\mathbb{C}[\mathrm{Gr}(3,12)]$ $r6$ dataset with penalty factor of 0.01, discouraging too many clusters.

in particular one which separates the SSYT which correspond to the cluster variables.

Overall K-Means managed to distinguish Grassmannian CV datasets using the rank partitioning, corroborating the PCA results, however struggled to separate the CV and NCV tableaux, further strengthening the successes of the supervised ML methods in learning this.

4.5. NN Gradient Saliency

The most promising results of the ML analysis are that simple NN architectures were able to determine whether a given SSYT corresponds to a cluster



(a) $\mathbb{C}[\mathrm{Gr}(3,12)]$, $r \leq 6$ (b) $\mathbb{C}[\mathrm{Gr}(4,10)]$, $r \leq 6$ (c) $\mathbb{C}[\mathrm{Gr}(4,12)]$, $r \leq 4$

Figure 6. NN gradient saliency images representing the averaged absolute values of the classification output gradients with respect to each of the respective tableaux inputs, for each of the Grassmannians considered. Lighter colours indicate the larger magnitude gradients, and hence the most dominantly useful entries for learning.

variable or not. Determining this is not possible directly, and the cluster variable and non-cluster variable datasets are certainly not distinguishable by eye. Moreover, unsupervised methods were also unable to identify simple structure which separates these datasets, strengthening further the performance of these NNs.

To dissect this exceptional performance, the technique of gradient saliency was used on the trained NNs to determine which parts of the inputs most significantly contributed to the respective classification of a tableaux throughout the test dataset. In this process, the gradient of the 0-dimensional, single entry, binary output is taken with respect to each of the input 24-dimensions, for all of the tableaux in the test dataset. These gradients are then averaged, absolute values taken, and plotted to provide a visual representation of the more dominant features used by the NNs to perform the classification. These images are shown for each of the binary classification investigations performed between cluster variable and non-cluster variable SSYT for each dataset in Figure 6. Note that since higher functionality was required for the NNs to perform the saliency analysis, `tensorflow` [1] was used to construct them (with the same hyperparameters as before).

In each of the images the lighter colours indicate the gradients with the larger average absolute values over the test dataset. Perhaps as expected for the $\mathbb{C}[\mathrm{Gr}(3,12)]$ dataset the bottom row has no dominant features as these entries are all an artefact of padding, as is the same for the $\mathbb{C}[\mathrm{Gr}(4,12)]$ rank 4 data where the last two columns are padded. Interestingly though, in all cases the central columns have equivalently negligible gradients, and hence negligible effects on the learning.

The most dominant features seem to be the top-right and bottom-left entries, excluding the $\mathbb{C}[\mathrm{Gr}(4,12)]$ rank 4 and $\mathbb{C}[\mathrm{Gr}(3,12)]$ rank 6 padding features. Hence the structure the NNs are using to discern whether a SSYT is a cluster variable or not is likely almost entirely determined by these entries. Some symbolic regression methods were implemented, using `gplearn`, to attempt to identify an equation that may relate these specific entries to the cluster variable prediction. However no suitable simple equation could be found, and hence the NNs use of these entries to determine the cluster variable nature of a generic tableaux in the Grassmannian is likely highly complicated.

5. Conclusion

In this paper, high performance computing (HPC) is applied to calculate cluster variables in Grassmannian cluster algebras $\mathbb{C}[\mathrm{Gr}(k,n)]$. We obtained cluster variables in $\mathbb{C}[\mathrm{Gr}(3,12)]$ up to degree 6 (the corresponding semistandard Young tableaux has at most 6 columns), in $\mathbb{C}[\mathrm{Gr}(4,10)]$ up to degree 6, and in $\mathbb{C}[\mathrm{Gr}(4,12)]$ up to degree 4. These cluster variables are computed for the first time and they have applications from geometry, to algebra, and to scattering amplitudes in physics [4, 20, 25, 28, 29]. Using these datasets, we verified Conjectures 3.1 and 3.2.

Supervised ML methods learnt to classify tableaux into each algebra with accuracy > 0.99, using the simple rank structure easily extractable from the data. These architectures then also learnt to identify cluster variable SSYT from tableaux which were not cluster variables for each algebra to accuracies ~ 0.95. This strong performance was further supported by PCA results showing the non-cluster variable data was representative of the true cluster variable data in each case. PCA also indicated that rank was the most dominant feature explaining data variation, due to the padding structure it requires. Clustering results with K-Means near perfectly separated the Grassmannians, but could also not differentiate the NCV tableaux, only clustering according to the rank information.

The lack of linear (and non-linear) structure in the datasets for the unsupervised methods to extract makes the supervised architecture even more impressive, confirming the utility of these advanced computational methods in analysing Grassmannian tableaux. Through methods of NN gradient saliency the dominant tableaux features used for learning were the last non-trivial entry of the first column and first entry of the last non-trivial column, and it is likely there is structure in these entries that strongly correlates with a SSYT being a cluster variable.

Acknowledgments

MWC would like to thank the Isaac Newton Institute for Mathematical Sciences, Cambridge, for support and hospitality during the programme 'Cluster algebras and representation theory' where work on this paper was undertaken. This work was supported by EPSRC grant no EP/R014604/1. PPD is grateful to the London Mathematical Society for grants 42035 and 42111, and to the London Institute for Mathematical Sciences for its hospitality. YHH would like to thank STFC for grant ST/J00037X/2. E. Heyes would like to thank SMCSE at City, University of London for the PhD studentship, as well as the Jersey Government for a postgraduate grant. E. Hirst would like to thank STFC for a PhD studentship. JRL is supported by the Austrian Science Fund (FWF): Einzelprojekte P34602. The computational results presented have been achieved in part using the Vienna Scientific Cluster (VSC), and the City, University of London high performance computing service (hyperion).

For the purpose of open access, the authors have applied a Creative Commons Attribution (CC BY) licence to any Author Accepted Manuscript version arising from this submission.

References

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems* (2015). Software available from tensorflow.org.

[2] G. Arias-Tamargo, Y.-H. He, E. Heyes, E. Hirst, and D. Rodriguez-Gomez, *Brain webs for brane webs*, Phys. Lett. B **833** (2022) 137376.

[3] N. Arkani-Hamed, J. Bourjaily, F. Cachazo, A. Goncharov, J. Trnka, and A. Postnikov, Grassmannian geometry of scattering amplitudes, Cambridge University Press (2016).

[4] N. Arkani-Hamed, T. Lam, and M. Spradlin, *Non-perturbative geometries for planar $N = 4$ SYM amplitudes*, J. High Energ. Phys. (2021), no. 03, 65.

[5] J. Bao, S. Franco, Y.-H. He, E. Hirst, G. Musiker, and Y. Xiao, *Quiver Mutations, Seiberg Duality and Machine Learning*, Phys. Rev. D **102** (2020), no. 8, 086013.

[6] J. Bao, Y.-H. He, and E. Hirst, *Neurons on Amoebae*, J. Symb. Comput. **116** (2022), 1–38.

[7] J. Bao, Y.-H. He, E. Hirst, J. Hofscheier, A. Kasprzyk, and S. Majumder, *Polytopes and machine learning*, (2021).

[8] ———, *Hilbert series, machine learning, and applications to physics*, Phys. Lett. B **827** (2022) 136966.

[9] K. Baur, D. Bogdanic, A. G. Elsener, and J.-R. Li, *Rigid indecomposable modules in Grassmannian cluster categories*, (2020).

[10] P. Berglund, B. Campbell, and V. Jejjala, *Machine learning Kreuzer-Skarke Calabi-Yau threefolds*, (2021).

[11] D. S. Berman, Y.-H. He, and E. Hirst, *Machine learning Calabi-Yau hypersurfaces*, Phys. Rev. D **105** (2022), no. 6, 066002.

[12] J. Carifio, J. Halverson, D. Krioukov, and B. D. Nelson, *Machine Learning in the String Landscape*, JHEP **09** (2017) 157.

[13] W. Chang, B. Duan, C. Fraser, and J.-R. Li, *Quantum affine algebras and Grassmannians*, Mathematische Zeitschrift **296** (2020), no. 3, 1539–1583.

[14] V. Chari and A. Pressley, *Factorization of representations of quantum affine algebras*, Modular Interfaces (Riverside CA 1995), AMS/IP Stud. Adv. Math **4** (1997), 33–40.

[15] V. Chari, A. Pressley, et al., A Guide to Quantum Groups, Cambridge University Press (1995).

[16] S. Chen, Y.-H. He, E. Hirst, A. Nestor, and A. Zahabi, *Mahler measuring the genetic code of Amoebae*, (2022).

[17] A. Cole, S. Krippendorf, A. Schachner, and G. Shiu, *Probing the structure of string theory vacua with genetic algorithms and reinforcement learning*, in 35th Conference on Neural Information Processing Systems, (2021).

[18] P.-P. Dechant, Y.-H. He, E. Heyes, and E. Hirst, *Cluster algebras: network science and machine learning*, (2022).

[19] L. J. Dixon, J. Drummond, T. Harrington, A. J. McLeod, G. Papathanasiou, and S. Marcus, *Heptagons from the Steinmann cluster bootstrap*, J. High Energ. Phys. (2017), no. 02, 137.

[20] J. Drummond, J. Foster, Ö. Gürdoğan, and C. Kalousios, *Tropical Grassmannians, cluster algebras and scattering amplitudes*, Journal of High Energy Physics **2020** (2020), no. 4, 146.

[21] S. Fomin and A. Zelevinsky, *Cluster algebras I: foundations*, Journal of the American Mathematical Society **15** (2002), no. 2, 497–529.

[22] S. Franco, D. Galloni, and A. Mariotti, *Bipartite field theories, cluster algebras and the Grassmannian*, J. Phys. A **47** (2014), no. 47, 474004.

[23] S. Franco, A. Hanany, Y.-H. He, and P. Kazakopoulos, *Duality walls, duality trees and fractional branes*, (2003).

[24] S. Franco and G. Musiker, *Higher cluster categories and QFT dualities*, Phys. Rev. D **98** (2018), no. 4, 046021.

[25] J. Golden, A. B. Goncharov, M. Spradlin, C. Vergu, and A. Volovich, *Motivic amplitudes and cluster coordinates*, Journal of High Energy Physics **2014** (2014), no. 1, 91.

[26] Y.-H. He, *Deep-learning the landscape*, (2017).

[27] Y.-H. He, E. Hirst, and T. Peterken, *Machine-learning dessins d'enfants: explorations via modular and Seiberg–Witten curves*, J. Phys. A **54** (2021), no. 7, 075401.

[28] N. Henke and G. Papathanasiou, *How tropical are seven-and eight-particle amplitudes*, Journal of High Energy Physics **2020** (2020), no. 8, 1–50.

[29] ———, *Singularities of eight- and nine-particle amplitudes from cluster algebras and tropical geometry*, Journal of High Energy Physics **2021** (2021), no. 7, 1–60.

[30] D. Hernandez and B. Leclerc, *Cluster algebras and quantum affine algebras*, Duke Mathematical Journal **154** (2010), no. 2, 265–341.

[31] E. Hirst, *Machine learning for Hilbert series*, in Nankai Symposium on Mathematical Dialogues: In Celebration of S. S. Chern's 110th Anniversary, (2022).

[32] V. Jejjala, D. K. Mayorga Pena, and C. Mishra, *Neural network approximations for Calabi-Yau metrics*, (2020).

[33] B. T. Jensen, A. D. King, and X. Su, *A categorification of Grassmannian cluster algebras*, Proceedings of the London Mathematical Society **113** (2016), no. 2, 185–212.

[34] D. Krefl and R.-K. Seong, *Machine learning of Calabi-Yau volumes*, Phys. Rev. D **96** (2017), no. 6, 066014.

[35] B. Leclerc, *Imaginary vectors in the dual canonical basis of $U_q(n)$*, (2002).

[36] M. Manko, *An upper bound on the critical volume in a class of toric Sasaki-Einstein manifolds*, (2022).

[37] G. Musiker and C. Stump, *A compendium on the cluster algebra and quiver package in Sage*, (2011).

[38] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, *Scikit-learn: Machine Learning in Python*, Journal of Machine Learning Research **12** (2011), 2825–2830.

[39] F. Ruehle, *Evolving neural networks with genetic algorithms to study the String Landscape*, JHEP **2017** (2017), no. 08, 038.

[40] J. S. Scott, *Grassmannians and cluster algebras*, Proceedings of the London Mathematical Society **92** (2006), no. 2, 345–380.

School of Mathematics, Kavli IPMU (WPI), UTIAS
The University of Tokyo, Kashiwa, Chiba, 277-8583, Japan
E-mail address: manwai.cheung@ipmu.jp

School of Mathematics, University of Leeds
Leeds, LS2 9JT, UK
Department of Mathematics, University of York
York, YO10 5DD, UK
York Cross-disciplinary Centre for Systems Analysis
University of York, York, YO10 5DD, UK
E-mail address: p.p.dechant@leeds.ac.uk

London Institute for Mathematical Sciences, Royal Institution
London, W1S 4BS, UK
Department of Mathematics, City, University of London
London, EC1V 0HB, UK
Merton College, University of Oxford
Oxford, OX1 4JD, UK
School of Physics, Nankai University
Tianjin, 300071, P. R. China
E-mail address: hey@maths.ox.ac.uk

Department of Mathematics, City, University of London
London, EC1V 0HB, UK
London Institute for Mathematical Sciences, Royal Institution
London, W1S 4BS, UK
E-mail address: elli.heyes@city.ac.uk

Department of Mathematics, City, University of London
London, EC1V 0HB, UK
London Institute for Mathematical Sciences, Royal Institution
London, W1S 4BS, UK
E-mail address: edward.hirst@city.ac.uk

Faculty of Mathematics, University of Vienna
Oskar-Morgenstern-Platz 1, 1090 Vienna, Austria
E-mail address: lijr07@gmail.com