LIMS-2022-012

Machine Learning Algebraic Geometry for Physics

Jiakang Bao[a,b] **Yang-Hui He**[b,a,c,d] **Elli Heyes**[a,b] **Edward Hirst**[a,b]

[a] *Department of Mathematics, City, University of London, EC1V 0HB, UK*
[b] *London Institute for Mathematical Sciences, Royal Institution, London W1S 4BS, UK*
[c] *Merton College, University of Oxford, OX1 4JD, UK*
[d] *School of Physics, NanKai University, Tianjin, 300071, P.R. China*

E-mail: jiakang.bao@city.ac.uk, hey@maths.ox.ac.uk, elli.heyes@city.ac.uk, edward.hirst@city.ac.uk

Abstract: We review some recent applications of machine learning to algebraic geometry and physics. Since problems in algebraic geometry can typically be reformulated as mappings between tensors, this makes them particularly amenable to supervised learning. Additionally, unsupervised methods can provide insight into the structure of such geometrical data. At the heart of this programme is the question of how geometry can be machine learned, and indeed how AI helps one to do mathematics.

This is a chapter contribution to the book *Machine learning and Algebraic Geometry*, edited by A. Kaspryzyk et al.

To the memory of Professor John K. S. McKay (1939-2022), with deepest respect.

Contents

1 Introduction

The ubiquitous interrelations between algebraic geometry and physics has for centuries flourished fruitful phenomena in both fields. With connections made as far back as Archimedes whose work on conic sections aided development of concepts surrounding the motion under gravity, physical understanding has largely relied upon the mathematical tools available. In the modern era, these two fields are still heavily intertwined, with particular relevance in addressing one of the most significant problems of our time - quantising gravity. String theory as a candidate for this theory of everything, relies heavily on algebraic geometry constructions to define its spacetime and to interpret its matter.

However, where new mathematical tools arise their implementation is not always simple. Current techniques for large scale statistical analysis would be at a loss without the recent exponential growth in the power of computation. It is hence only natural to investigate the prosperity of big data techniques in guiding this innate partnership between geometry and physics; and for this we focus on a big data key player: machine learning (ML).

Machine learning has seen a vast array of applications to algebraic geometry in physics, with first introduction to the string landscape in [1–5], and numerous current uses in finding suitable metrics on the landscape [6–15, 15–20]. More generally, the question of machine-learning mathematics and using data scientific methods as well as artificial intelligence to help humans to do mathematics is very much in the air [21–27].

Many of these methods use neural networks (NN) in some form as generalisations of the multi-layer perceptron (MLP) for non-linear function approximation. In any respect, new approaches to mathematical physics will continue to rely further on techniques that can suitably handle large amounts of data. Pattern recognition for conjecture formulation may even eventually become more efficient with ML techniques through clustering and degree of freedom identification with methods such as principal component analysis (PCA).

In this work we review a selection of contemporary studies into the use of ML in algebraic geometry for physics. Specifically each section finds focus on application to: §2 hypersurfaces [28], §3 polytopes [29], §4 Hilbert series [30], §5 amoebae [31], §6 brane webs [32], §7 quiver mutation [33], §8 dessins d'enfants [34], and §9 Hessian manifolds.

The string landscape Superstring theory requires the spacetime to be of (real) dimension 10. To reconcile with our experience of 4-dimensional spacetime, the extra dimensions must be curled up so that they cannot be observed at low energy. In other words, our theory is compactified on a 6-dimensional manifold $\mathcal{M}_6$ where the full spacetime is $\mathcal{M}_{10} = \mathbb{R}^{3,1} \times \mathcal{M}_6$. In fact, $\mathcal{M}_6$ can be endowed with complex structure, and we shall henceforth refer to it as a threefold in terms of its complex dimension. As pointed out in [35], a standard solution for this threefold is a Ricci flat Kähler manifold. Such manifolds are famously known as the Calabi-Yau (CY) manifolds.

The physics in the Minkowski spacetime $\mathbb{R}^{3,1}$ is dictated by the geometry and topology of the CY threefolds. It is still not clear which CY_3 is *the* manifold for string theory compactification. Nevertheless, people have made great efforts in finding distinct CY threefolds

over the past few decades. Nowadays, the full list contains over billions of possible candidates. In the era of big data, the technique of machine learning therefore naturally enters this string landscape. For recent summaries and reviews, see [21, 25, 27, 36–38].

We should emphasize that the concept of CY n-fold in this note is defined in a rather broad sense. We shall not only discuss compact CYs, but also take non-compact (also known as "local") ones into account. Physically, one can turn on the Ω-background which localizes our theory on the non-compact CY and effectively compactifies it to 4d. Moreover, we will also consider singular algebraic varieties, especially Gorenstein singularities which can be resolved to smooth CYs.

Machine Learning Most machine learning algorithms can be put into two categories: unsupervised learning and supervised learning methods (there are other categories such as reinforcement learning and semi-supervised learning). Unsupervised methods are used when one is given unlabelled data $\{x_i\}$ and wishes to find patterns/structures within the data. These methods can be broken down further into clustering methods, such as k-means clustering and dimensionality reduction methods, such as principal component analysis (PCA). Supervised learning on the other hand, is used when one has labeled data $\{(x_i, y_i)\}$ and wishes to approximate a mapping function $f : x_i \mapsto y_i$, such methods include linear regression, neural networks (NN), of which convolutional neural networks are a subtype (CNN), random forests, and support vector machines. Supervised learning can also be broken down further into regression and classification problems. In Regression problems the output variables y are continuous numerical values whereas in classification problems y is discrete and not necessarily numerical. For regression problems common performance measures include mean absolute error (MAE) and mean squared error (MSE), for classification problems accuracy is often used and Matthew's Correlation Coefficient (MCC) is another popular choice where the classes are of different sizes.

In order to get an unbiased evaluation of how well your supervised learning algorithm is learning to map the data, the data is shuffled and then separated into training and test sets. The training dataset is used to fit the model parameters, and the test dataset is used to provide an unbiased evaluation of the model. The usual train:test split is 80:20. Cross-validation is another method used to obtain an unbiased evaluation of the model performance, whereby the data is first shuffled and split into k groups, then each group is taken in turn to be the test set and the remaining groups are combined to create the training set. In each case the model is trained on the training set, evaluated on the test set, and the evaluation score is recorded. The mean and standard deviation of the k evaluation scores are then used to determine the model performance.

For the projects reviewed here, most work was carried out in `Python` with the use of `TensorFlow` [39] and `scikit-learn` [40] for machine learning implementation, `Yellowbrick` [41] for visualisations and `ripser` [42] for topological data analysis.

2 Calabi-Yau Hypersurfaces

There have been many very interesting applications of machine learning to the Calabi-Yau landscape. Here we focus on summarising the work in [28] which centres its attention on

a specific subset of the landscape: Calabi-Yau 3-folds constructed from hypersurfaces in weighted projective space, $\mathbb{P}^4$.

One of the first big datasets in theoretical physics came from a generalisation of the quintic 3-fold in $\mathbb{P}^4$. The generalisation assigned weights to the respective identification used in defining the projective space, such that for some $\mathbb{C}^{n+1}$ with coordinates $\{z_1, z_2, ..., z_{n+1}\}$, a set of $n+1$ coprime integer weights, w_i, could be selected to describe the identification

$$(z_1, z_2, ..., z_{n+1}) \sim (\lambda^{w_1} z_1, \lambda^{w_2} z_2, ..., \lambda^{w_{n+1}} z_{n+1}) \,, \tag{2.1}$$

$\forall \lambda \in \mathbb{C}$. These, now weighted, projective spaces can have codimension-1 hypersurfaces drawn within them, and it turns out for only a specific set of 7555 weight combinations, for the $n = 4$ case, can one define hypersurfaces within them which are Calabi-Yau in nature [43]. This construction method for Calabi-Yau 3-folds now forms a distinguished subset of the full Kreuzer-Skarke database [44], which describes Calabi-Yau hypersurfaces in more general toric varieties.

Why only these specific 7555 weight combinations permit Calabi-Yau hypersurfaces is perplexing. Beyond the defining Calabi-Yau feature of vanishing first Chern class causing the hypersurface polynomial to have degree $\sum_i w_i$, there is a necessary but not sufficient condition on the weights for the hypersurfaces to be consistently defined over the singular sets created by the projective identification. This condition is *transversity* and sets a specific divisibility condition on the weights.

To examine this database the work of [28] turned to tools from supervised and unsupervised ML, which we summarise in this section. Unsupervised methods sought to provide general data analysis of this special database; whilst supervised methods aimed to learn the topological properties of the hypersurfaces directly from the weights. The learning hence emulating the highly non-trivial formulas from the Poincaré polynomial, $Q(u, v)$, expansion which provides the Hodge numbers, $h^{\alpha,\beta}$, and a direct equation for the Euler number, χ.

$$\begin{aligned} Q(u,v) &= \sum_{\alpha,\beta} h^{\alpha,\beta} u^\alpha v^\beta = \frac{1}{uv} \sum_{l=0}^{\sum_i (w_i)} \left[\prod_{\tilde{\theta}_i(l) \in \mathbb{Z}} \frac{(uv)^{q_i} - uv}{1 - (uv)^{q_i}} \right]_{\text{int}} \left(v^{\text{size}(l)} \left(\frac{u}{v} \right)^{\text{age}(l)} \right) , \\ \chi &= \frac{1}{\sum_i (w_i)} \sum_{l,r=0}^{\sum_i (w_i) - 1} \left[\prod_{i | l q_i, r q_i \in \mathbb{Z}} \left(1 - \frac{1}{q_i} \right) \right] , \end{aligned} \tag{2.2}$$

such that normalised weights $q_i = w_i / \sum_i (w_i)$ are operated on with multiple divisibility checks and other complicated computation steps involving $\text{age}(l) = \sum_{i=0}^{4} \tilde{\theta}_i(l)$ and $\text{size}(l) = \text{age}(l) + \text{age}(\sum_i (w_i) - l)$; where in $(\mathbb{R}/\mathbb{Z})^5$, $\tilde{\theta}_i(l)$ is the canonical representative of $l q_i$ [45, 46].

2.1 PCA, Clustering, and TDA

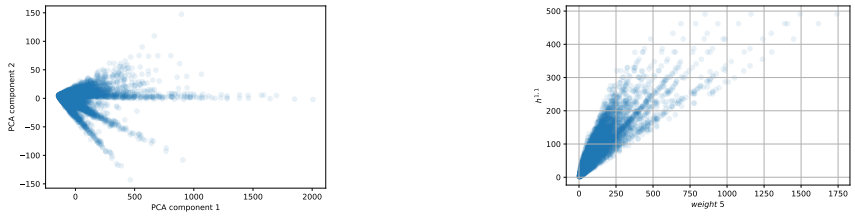
The dataset of weighted-$\mathbb{P}^4$'s which admit Calabi-Yau hypersurfaces takes the form of 7555 sorted 5-vectors of positive integer weights. In addition to this information, the non-trivial

Hodge numbers $\{h^{1,1}, h^{2,1}\}$ of each hypersurface are provided with these weights of the ambient projective variety in the Kreuzer-Skarke database: http://hep.itp.tuwien.ac.at/~kreuzer/CY/.

For dataset feature comparison, and to benchmark learning performance, three further datasets were generated of: 1) Random positive integers, 2) Coprime positive integers, 3) Transverse coprime integers. Therefore with the Calabi-Yau weight data these datasets form a series, each with more of the necessary (but not sufficient) conditions for the 'Calabi-Yau property'. Furthermore these datasets were constructed so as to have no overlap with each of the other datasets in the series, therefore not satisfying the properties at the next level and hence ensuring artefacts of the analysis could be attributed to the property inclusion.

Principal Component Analysis To determine structure in the weight vector data, PCA was applied to each of the datasets. This method finds the linear combinations of the weights which dictate the most dominant variation in the data. The process diagonalises the covariance matrix such that eigenvalues give successively lower variances and the respective eigenvectors are the principal components which best describe the distribution of the weights.

Since the weight vectors are sorted in increasing order, the principal components have a vague correlation with the weights themselves, however as the PCA plot in Figure 2.1a shows the Calabi-Yau data has a clear linear clustering structure in the first two principal components, not present in the other generated datasets.



(a) 2d PCA (b) w_5 vs $h^{1,1}$

Figure 2.1: Linear clustering behaviour shown from both the 2d PCA projection of the dataset (a), and the distribution of largest ambient space weight against $h^{1,1}$ of the hypersurface (b).

Clustering Plotting the weights of each ambient space against the Hodge numbers of the Calabi-Yau hypersurface exhibits a familiar behaviour to that seen in the PCA, as shown in 2.1b. The linear clustering behaviour observable by eye suggests a surprising simpler structure in the data which ML methods can make use of for approximate learning. However prior to examining the supervised learning in 2.2, the clustering behaviour can be quantitatively verified with the use of the unsupervised method K-Means clustering.

K-Means clustering takes as input a number to define how many clusters to compute, then iteratively updates the cluster centres until the average distance between any datapoint and its nearest cluster centre is minimised. The number of clusters was determined using a variant of traditional elbow methods, recommending an optimum of 10 clusters for

the clustering algorithm. This distance measure is known as *inertia*, and an equivalent version normalised by the number of clusters and datapoints gives a final metric to establish how suitable this clustering behaviour was.

For this 2d w_5 vs $h^{1,1}$ data, to evaluate the linear behaviour each datapoint was projected onto its ratio $h^{1,1}/w_5$, and then clustering performed on this now 1d data. The final normalised inertia took value: 0.00084; which may be interpreted as after determining optimal cluster centres each datapoint is on average less than 0.1% of the full data range from its nearest cluster centre. These are exceptionally good clustering results and strongly corroborate the behaviour observed.

Topological Data Analysis Persistent homology is a tool form topological data analysis most useful for analysis of higher-dimensional data structures that cannot be plotted directly. Since the weight data here is 5 dimensional it is perfectly suited for this line of analysis also. For other uses of TDA in the theoretical physics please see [47, 48].

The process involves the plotting of the 5d data in $\mathbb{R}^5$, and drawing 5d balls centred on each point all with radius d. As the ball radii are varied from $0 \mapsto \infty$ they begin to intersect, a n-simplex is then drawn between n points when their balls intersect; note in 5d only up to 5-simplices can be drawn. This forms a chain of Vietoris-Rips complexes for each d value which changes the number of intersections.

The homology of this chain can then be computed, with the use of the `python` library `ripser` [42]; and due to computational memory restrictions we limit analysis to H_0 and H_1 in this study. H_0 features are each of the points individually, hence all 'born' at $d = 0$, and die as they become connected components (the usual convention for the feature allocated to the smaller component dying as two become connected). These features hence provide useful information on the distribution of datapoints. Conversely H_1 features are sets of 1-simplices (edges) that form the boundary of a union of 2-simplices not in the complex. They keep track of 2d holes in the data and gives further useful information about the datapoint distribution.

The H_i features are then plotted as (birth,death) pairs on a persistence diagram, for the Calabi-Yau weight data this is shown in Figure 2.2. The H_0 features show gaps in the line, indicating that some datapoints are separated in some way into clusters which join together at higher d values. This behaviour is expected since some datapoints have particularly high w_5 values, causing them to be significantly further away from the bulk of the data, likely connecting amongst themselves along the linear clusters before joining the bulk complex. The H_1 features all lie close to the diagonal, behaviour typical of noise, and thus there is not any further higher-dimensional structure in the data connecting these classes together to form persistent loops. Both these result support the observed clustering behaviour.

2.2 Neural Networks for Hypersurfaces

For the supervised learning of this data, two investigations were carried out, one using regressors to predict topological parameters of the hypersurface, and the other classifiers

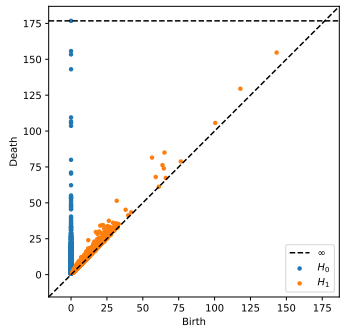


Figure 2.2: Persistent diagram for the H_0 and H_1 homology groups of the CY data's Vietoris-Rips complex filtration.

to distinguish weight vectors that admit Calabi-Yau hypersurfaces from those which do not.

Regression The first used a typical feed-forward neural network regressor to learn the Hodge numbers and Euler number from the 5 vectors of weights, using the standard mean squared error loss. For Calabi-Yau 3-folds there is a simple expression for the Euler number in terms of the Hodge numbers: $\chi = 2(h^{1,1} - h^{2,1})$. Performance was best measured on the independent test data using the R^2 metric, determining the performance of the network with respect to a null model always predicting the mean output. The metric evaluates in the range $(-\infty, 1]$ with value 1 for perfect prediction. To provide confidence in the measure 5-fold cross-validation was also performed such that 5 identical architectures were independently trained and tested on 5 different partitions of the data.

The results for this investigation are given in Table 2.1. They show very strong learning with R^2 values close to 1. Since the formulas 2.2 require a lot of divisibility checks which NNs are notoriously poor at, it is likely the networks are making use of some other structure, perhaps the linear clustering observed, to predict these topological parameters.

Measure	Property			
	$h^{1,1}$	$h^{2,1}$	$[h^{1,1}, h^{2,1}]$	χ
R^2	0.9630 $\pm$ 0.0015	0.9450 $\pm$ 0.0133	0.9470 $\pm$ 0.0041	0.9510 $\pm$ 0.0023

Table 2.1: Learning the non-trivial Hodge numbers and Euler number from the Calabi-Yau 5-vectors of weights. Measurement of learning performance uses 5-fold cross-validation to provide an average and standard error on each measure's value.

Classification The second investigation used a variety of classification architectures (including NNs), most notably performance was not significantly improved on by the more complicated architectures compared to the simple prototypical classification architecture of Logistic Regression. Distinguishing each of the generated datasets of weights from the Calabi-Yau dataset of weights gave accuracies ~ 0.7 in all cases, showing some learning, particularly impressive for the case where both datasets exhibited transversity.

However these accuracy scores are not at the level of usual successful machine learning investigations. Hence to probe this further the misclassifications of the trained logistic regressor predicting on all the Calabi-Yau data were examined. These results plotted against their respective $(h^{1,1}, h^{2,1})$ values showed a clear correlation where pre-training against random data only misclassified at low $h^{2,1}$ values, and pre-training against transverse data only misclassified at low $h^{1,1}$. This behaviour is shown in Figure 2.3.

(a) Train with Random

(b) Train with Transverse

Figure 2.3: The Logistic Regressor trained for binary classification with the Calabi-Yau weights and the: (a) Random weights, (b) Transverse weights; predicting all the Calabi-Yau weights to be Calabi-Yau or not. The classification results are plotted with respect to the hypersurface's non-trivial Hodge numbers, showing perfect prediction above certain bounds for $h^{2,1}$ or $h^{1,1}$ respectively.

Further investigations then binned the data according to the Hodge numbers, and trained/tested within each bin. Interestingly, which weights took dominant importance flipped between the $h^{1,1}$ and $h^{2,1}$ binning investigations.

3 Polytopes and Toric Geometry

For local/non-compact CY $(d+1)$-folds, their information can be combinatorially encoded by convex polytopes on the lattice $N := \mathbb{Z}^d$. Roughly speaking, one constructs the $(d+1)$-dimensional cone generated by the polytope. Then the maximal spectrum of the group algebra generated by its dual cone yields a Gorenstein singularity, which can be resolved to a local CY $(d+1)$-fold whose compact base is the Gorenstein Fano toric d-fold specified by the polytope. In other words, considering the fan of the polytope and the d-dimensional cones therein, we can construct a projective d-dimensional variety by gluing the affine varieties from these cones. In this section, we shall mainly focus on the polytopes corresponding to toric Fano varieties.

Definition 3.1. *A polytope P is Fano if the origin $\mathbf{0}$ is strictly in its interior* $\mathrm{int}(P)$ *and all its vertices are primitive*[1]*. It is further called a canonical Fano polytope if* $\mathrm{int}(P) \cap \mathbb{Z}^d = \{\mathbf{0}\}$.

It is known that there are 16 toric Fano 2-folds with at worst canonical singularity which are precisely the reflexive polygons. For 3-folds, there are 674688 of them[2] as classified in

[1] A lattice point $\boldsymbol{v} = (v_1, \ldots, v_d)$ is primitive if $\gcd(v_i) = \pm 1$.

[2] In contrast, there are only 4319 reflexive polyhedra due to the extra criterion that not only is there a single interior point but also all bounding facets/edges are distance 1 to this point.

[49]. We will then deal with those canonical Fano 3-folds in our machine learning problem while general Fano varieties are considered for $d = 2$.

The set of vertices of a lattice polytope is a natural option as our input. However, as pointed out in [29], the vector of Plücker coordinates as a geometric invariant theory representation is a better choice for learning various properties of the polytope.

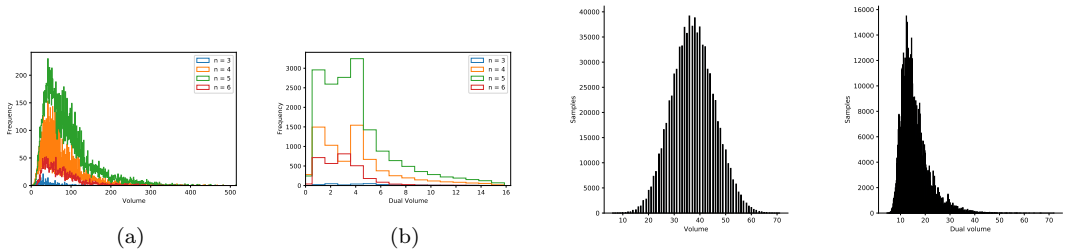
Definition 3.2. *Let P be a d-dimensional polytope with n vertices. This structure can be reformatted into an $d \times n$ matrix V where each column corresponds to a vertex $\boldsymbol{v}_i$. Consider the integer kernel of V (aka grading) and take the maximal minors of* $\ker(V)$*; this gives a list of integers known as the Plücker coordinates, which we regard as a point in projective space.*

The Plücker coordinates reveal linear relations among vertices, and it is not hard to see that they are invariant under $\mathrm{GL}(d, \mathbb{Z})$ transformations. However, this also gives rise to the ambiguity representing a unique polytope P. One may introduce the quotient gradings to further distinguish those polytopes. Here, we shall instead impose the condition that the vertices of P generate the lattice $\mathbb{Z}^d$. This uniquely determines P.

Moreover, the initial ordering of the n vertices leads to $n!$ equivalent ways of writing the Plücker coordinates. This is due to the symmetric group $\mathfrak{S}_n$ permuting the vertices. Such redundancy is irrelevant for Plücker coordinates representing a polytope, but this can be used to enhance our datasets. We will choose 3 and 10 distinct Plücker coordinate orderings for each polygon and polyhedron respectively in our data.

3.1 Volumes and Dual Volumes

Give a polytope, two straightforward quantities one can obtain are its volume and dual volume (i.e., the volume of its dual polytope). In particular, we take the normalized volume where each simplex has volume 1. For $d = 2$, we generate random Fano polygons with the numbers of vertices $n \in \{3, 4, 5, 6\}$. The volumes and dual volumes take values within the range $[3, 514]$ and $[0.21, 15.37]$ respectively. Their distributions are shown in Figure 3.1 (a, b). For each n, a network is trained separately.



(a) (b) (c) (d)

Figure 3.1: Distributions of the (a) volumes and (b) dual volumes of 2d polygons and distributions of the (c) volumes and (d) dual volumes of 3d polyhedra.

We can perform the similar experiment for 3d polytopes as well. Since there is sufficient data, the randomly chosen polyhedra are restricted to be canonical Fano and we will not

train different networks for different n as in the 2d case. The distributions of data are given in Figure 3.1 (c,d).

Here, we apply MLP to this regression problem[3]. As the Plücker coordinates for polytopes can have different lengths, we find that padding zeros to the input vectors would give best performance. The results are summarized in Table 3.1. We may also use the Plücker coordinates as input to predict other properties of the polytopes. More details can be found in [29].

Polygons (frequency)	Triangles (277)	Quadrilaterals (7041)	Pentagons (16637)	Hexagons (3003)
Output	Volume			
MAE	0.209	0.625	1.051	3.359
Accuracy	1.000	1.000	1.000	0.997
Output	Dual Volume			
MAE	1.181	0.642	0.818	0.941
Accuracy	0.501	0.754	0.638	0.557
Polyhedra (frequency)	Toric Canonical Fano Polyhedra (780000)			
Output	Volume		Dual Volume	
MAE	1.680		2.590	
Accuracy	0.936		0.890	

Table 3.1: The machine learning results for volumes and dual volumes. The "Accuracy" for polygons is the accuracy$\pm 0.05 \times$ range. Likewise, for those involving polyhedra, the "Accuracy" is the accuracy± 4. The frequency stands for the number of samples trained in each model.

3.2 Data Projection and Visualization

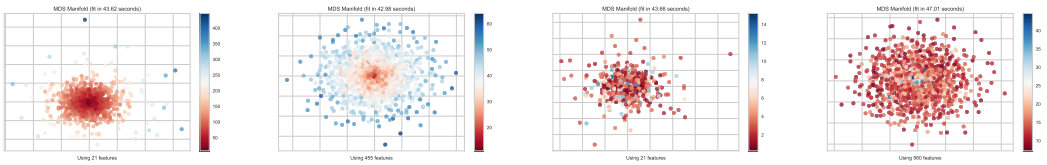
To study the potential structures captured by the machine, one may try to project the input data to lower dimensions. A handy way to perform this unsupervised learning technique is the so-called manifold learning [50]. For instance, the multi-dimensional scaling (MDS)[4] projects a Plücker input vector $\boldsymbol{p} = (p^0 : \cdots : p^{l-1})$ of length l to some vector $x = (x^0, \ldots, x^{k-1})$ with $k < l$ by minimizing the cost function (aka stress)

$$S = \left(\sum_{i<j\leq N} (D_{ij} - d_{ij})^2 \right)^{1/2} , \tag{3.1}$$

where N is the number of samples and D_{ij} (d_{ij}) denotes the Euclidean distance between $\boldsymbol{p}_i$ ($\boldsymbol{x}_i$) and $\boldsymbol{p}_j$ ($\boldsymbol{x}_j$). The MDS projections for polygons and polyhedra are shown in Figure 3.2 (a, b) with the hue based on volumes.

[3]The detailed structures of the networks can be found in [29]. Slight changes in the networks should not affect our results in mathematical problems [25].

[4]Readers are referred to [51] for an introduction to MDS.



(a) (b) (c) (d)

Figure 3.2: The MDS projections of Plücker coordinates. The volumes are given in (a) polygons and (b) for polyhedra. The dual volumes are given in (c) polygons and (d) for polyhedra. Different colours of data points indicate different (dual) volumes. Here, we use 1000 random samples for each plot as an illustration.

As we can see, the vectors $\boldsymbol{x}$ are scattered around a centre point. The closer $\boldsymbol{x}$ is to the centre, the smaller volume the corresponding polytope would have. Since this is a mathematical fact, the number of samples should not affect the results and minimizing S is therefore equivalent to the goal of finding $\boldsymbol{x}$ such that $D_{ij} - d_{ij} \approx 0$ for any i, j. As the positions of the data points with different volumes mainly depend on the distances from the centre point in Figure 3.2 (a, b), 1-dimensional projections are actually sufficient as shown in [29]. We are using the 2-dimensional plots here for later comparison with dual volumes. From the MDS projections, it seems to indicate a map $f : \boldsymbol{p} \mapsto x$ such that Vol $\sim x + 1$. This should be in line with the fact that Vol $= \sum_{\alpha \in \mathcal{V}} p^\alpha$ where $\mathcal{V} \subseteq \{0, \ldots, l-1\}$. More detailed analysis can be found in [29].

Now let us apply MDS to dual volumes. As visualized in Figure 3.2 (c, d), the points with different colours are distributed more randomly. This also explains why the machine learning performance on dual volumes is inferior to the one on volumes.

4 Hilbert Series

Hilbert series (HS) are important tools in modern geometry. Their use in physics centres in supersymmetric quantum field theories where they are useful for BPS operator counting [52], as well as in more general string theory where they provide geometric information about a theory's moduli space [53].

Mathematically, HS encode embedding-specific information about a complex projective variety X, embedded in $\mathbb{P}_{\mathbb{C}}(p_0^{q_0}, \ldots, p_s^{q_s})$, denoted where weight p_i occurs q_i times. The embedding induces a grading on its coordinate ring $R = \mathbb{C}[X_0, \ldots, X_k]/I$, such that $R = \bigoplus_{i \geq 0} R_i$; for variables X_i and I the homogeneous ideal generated by the polynomials defining the variety [54]. Now the HS is the generating function for the dimensions of the graded pieces of R,

$$H(t; X) = \sum_{i=0}^{\infty} (\dim_{\mathbb{C}} R_i) t^i \; ; \tag{4.1}$$

such that $\dim_{\mathbb{C}} R_i$ is also the number of independent degree i polynomials on the variety.

Via the Hilbert-Serre theorem [55], the HS can be written in two forms:

$$(1)\quad H(t;X)=\frac{P(t)}{\prod_{i=0}^{s}(1-t^{p_i})^{q_i}},\quad (2)\quad H(t;X)=\frac{\tilde{P}(t)}{(1-t^{j})^{dim+1}}, \tag{4.2}$$

for polynomials P and $\tilde{P}$ such that the variety has Gorenstein index j and dimension dim. In addition if the numerator polynomial $\tilde{P}$ is palindromic the variety is Gorenstein in nature [56].

The work in [30, 57] initiates the application of machine learning techniques to these objects, with the aim that from the first terms of the series' Taylor expansions one can extract information about the underlying variety from the HS closed forms from (4.2). With this information one can then compute all higher terms, and hence information about any order BPS operators from just information on the first few, as well information about the physical theory's moduli space.

4.1 Gathering HS Data

The data used in the investigations included HS associated to algebraic varieties, retrieved from the GRDB [58, 59]. These used a database of candidate HS conjecturally associated to three-dimensional $\mathbb{Q}$-Fano varieties with Fano index one, as constructed in [60, 61]. In addition data was also generated by sampling the hyperparameters of the HS form (4.2) from distributions fitted from the GRDB data; ensuring that some physical conditions were met to make the data representative and that there was no overlap with the GRDB data.

For both HS datatypes, which we call 'real' and 'fake' respectively, the HS closed forms are Taylor expanded to provide coefficient lists. The coefficient lists are hence vectors of integers, and make up the ML input for the investigations carried out. Two coefficient vectors were used as inputs, one from the start of the series (coefficients 0-100) and one from deeper into it (1000-1009). Using the higher order terms in a separate vector as well allows one to assess the importance of orbifold points which take effect deeper in the series leading to the coefficient growth rate being more stable.

4.2 Learning Geometric Properties

In each of the five investigations performed with this data different properties of the HS were learnt. NN regressors were used to predict the vector of embedding weights (p_i), whilst classifiers learnt the Gorenstein index j and dimension dim directly. Further to this three binary classification investigations were performed to evaluate whether HS were Gorenstein, were complete intersections, or whether they were from the GRDB/generated by us.

Architectures For all investigations the NNs used had 4 dense layers of 1024 neurons and 0.05 dropout, each with ReLU activation. Training was in batches of 32 for 20 epochs using the Adam optimiser to minimise either the log(cosh) or cross-entropy loss functions for regression or classification respectively. 5-fold cross-validation was used to provide confidence for the metrics used to evaluate the learning. For regression this was assessed with the MSE metric, and for classification accuracy and MCC were used.

Investigation	Orders of Input	Performance Measure	
		MAE	
Weights (p_i)	0-100	1.94 ± 0.11	
	1000-1009	1.04 ± 0.12	
-	-	Accuracy	MCC
Gorenstein Index j	0-100	0.934 ± 0.008	0.916 ± 0.010
	1000-1009	0.780 ± 0.018	0.727 ± 0.022
Dimension dim	0-100	0.995 ± 0.005	0.993 ± 0.006
	1000-1009	0.865 ± 0.024	0.822 ± 0.031

Table 4.1: Results for learning parameters of the full Hilbert series from vectors of coefficients at specified orders. Learning embedding weights of Form (1) in (4.2) used NN Regressors, whilst learning Gorenstein index and dimension from Form (2) in (4.2) used NN classifiers.

Investigation	Orders of Input	Performance Measures	
		Accuracy	MCC
Gorenstein	0-100	0.844 ± 0.087	0.717 ± 0.155
	1000-1009	0.954 ± 0.043	0.919 ± 0.073
Complete Intersection	0-100	0.762 ± 0.010	0.544 ± 0.030
	0-300	0.951 ± 0.005	0.902 ± 0.010
GRDB	0-100	1.000 ± 0.000	1.000 ± 0.000
	1000-1009	1.000 ± 0.000	1.000 ± 0.000

Table 4.2: Binary classification results for distinguishing whether input vectors of HS coefficients of specified orders come from full HS which represent Gorenstein varieties, complete intersections, or whether the HS was from the GRDB or generated as 'fake' data.

Results Results for each of the investigations are given in Tables 4.1 and 4.2. The regressor results show that the weights of the ambient space could be learnt to the nearest integer, performing better with higher orders. The geometric parameters were learnt with exceptional accuracy also, but conversely performed better using lower order coefficients.

The binary classification results in Table 4.2 show the Gorenstein and complete intersection properties could be confidently detected from higher orders, whilst the true GRDB vs fake generated data was trivially distinguished, as also corroborated by PCA, this indicates how difficult it is to generate representative HS data.

5 The Genus of Amoebae

Given a Newton polynomial $P(z, w)$, its amoeba is the set[5]

$$\mathcal{A}_P := \{(\log|z|, \log|w|) \in \mathbb{R}^2 : P(z, w) = 0\}. \tag{5.1}$$

Amoeba is an important concept in tropical geometry [62–64] and has many applications in physics [65–69]. Roughly speaking, an amoeba is the thickening of the dual web of its

[5]Amoebae can certainly be defined for any Laurent polynomial $P(z_1, \ldots, z_r)$. Nevertheless, we shall focus on $r = 2$ in this section.

associated toric diagram. It is often not easy to determine the boundary of an amoeba for a generic Newton polynomial. In [70], the idea of lopsidedness was applied to find the boundaries of any amoebae. We say a list of positive numbers is lopsided if one of the numbers is greater than the sum of all the others. In particular, a list $P\{z,w\}$ can be constructed from $P(z,w)$ as $P\{z,w\} = \{|m_i(z,w)|\}$ where $m_i(z,w)$ are the monomials of $P(z,w)$. This yields the so-called lopsided amoeba $\mathcal{LA}_P = \{(z,w)|P\{z,w\} \text{ is not lopsided}\}$. This provides an approximation of the amoeba and its boundary in the sense that $\mathcal{LA}_P \supseteq \mathcal{A}_P$. To get better approximations, we may consider the cyclic resultant of $P(z,w)$, or equivalently,

$$\widetilde{P}_n(z,w) := \prod_{k_1=0}^{n-1}\prod_{k_2=0}^{n-1} P\left(\mathrm{e}^{2\pi i k_1}z, \mathrm{e}^{2\pi i k_2}w\right). \tag{5.2}$$

Then $\mathcal{LA}_{\widetilde{P}_n}$ converges uniformly to $\mathcal{A}_P$ as $n \to \infty$. In other words, we can get better approximations of the amoeba by taking higher level n.

As a result, the complementary regions of an amoeba on $\mathbb{R}^2$ would also be determined once we know its boundary. In particular, the number of bounded complementary regions is called the genus of the amoeba. As a matter of fact, the maximal possible genus is equal to the number of internal points of the corresponding lattice polygon. Machine learning techniques were used in [31] to find the genus of an amoeba when varying the coefficients of $P(z,w)$.

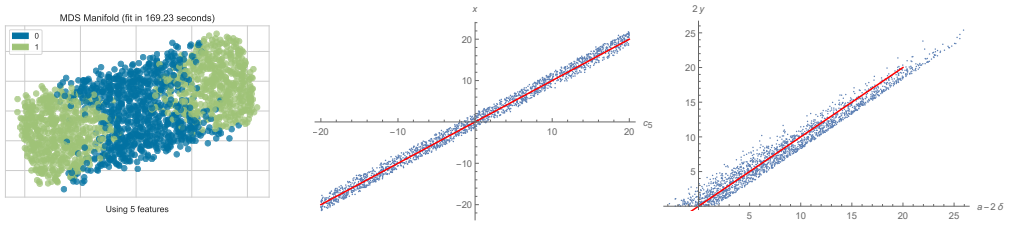
Let us illustrate this with one of the simplest examples, that is, F_0 whose Newton polynomial reads $P = c_1 z + c_2 w + c_3 z^{-1} + c_4 w^{-1} + c_5$. It is not hard to see that the associated polygon has one single internal point. Therefore, the genus g is either 0 or 1, and we simply have a binary classification problem. Moreover, we shall mainly focus on the lopsided amoeba $\mathcal{LA}_P$, that is, the approximation at $n = 1$.

5.1 Data Projection and Visualization

Using the coefficients $\{c_1, \ldots, c_5\}$ (sampled $c_{1,2,3,4} \in [-5,5]$, $c_5 \in [-20,20]$) as input and g as output, a simple MLP can easily reach over 0.95 accuracy with fewer than 2000 training samples. We will discuss the structure of the network in the next subsection while here let us first apply some unsupervised techniques to understand what information we might be able to extract from the machine.

To analyze relevant features of the problem, let us again apply MDS projection as visualized in Figure 5.1(a). As we can see, the blue ($g = 0$) and green ($g = 1$) points have a rather clear separation. To understand this scattering plot, let us analyze the two coordinates x and y separately.

In Figure 5.1(b), we plot x against c_5. The red line $x = c_5$ shows a good fit for the points. We may therefore conclude that one of the coordinates in the projection is simply the coefficient c_5. It is then natural to expect that y would be transformed from the remaining four coefficients. By curve fitting, we find that $|y_{\text{fit}}| = 0.141|c_1 c_3| + 0.166|c_2 c_4| + 2.411$. Astute readers may have already found that this is very close to the approximation of square roots. Indeed, let $a := 2|c_1 c_3|^{1/2} + 2|c_2 c_4|^{1/2}$. Then it can be approximated as $a/2 \approx 0.1|c_1 c_3| + 1.2 + 0.1|c_2 c_4| + 1.2$. In Figure 5.1(c), we plot $2|y|$ versus $a - 2\delta$ where



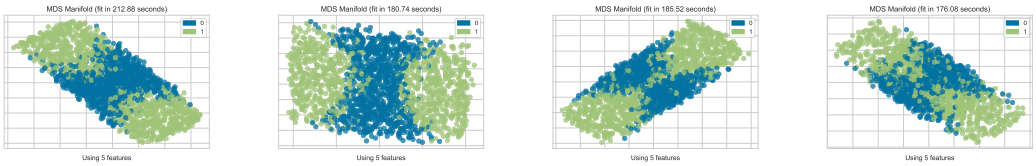
(a) (b) (c)

Figure 5.1: The MDS manifold projection $(c_1, \ldots, c_5) \mapsto (x, y)$ and the analysis of (x, y).

$\delta = |y_{\text{fit}}| - |y|$. Again, we find that the points scatter along the 45-degree line crossing the origin. Hence, the other transformed coordinate in the data projection actually indicates $|y| \approx a/2$.

Therefore, the genus should be encoded by two parameters: c_5 and a. Indeed, applying the lopsidedness condition, we find that $g = 0$ if and only if $|c_5| \leq a$. This in fact agrees with the scattering plot in Figure 5.1(a) where the separation of blue and green regions, albeit not perfectly precise, approximates the lines $|y| = |x|$.

One may also try to apply such projection to higher n. As shown in Figure 5.2, the plots still have the similar scatterings to the one for $n = 1$.



(a) (b) (c) (d)

Figure 5.2: The MDS manifold projections for the datasets with (a) $n = 2$, (b) $n = 3$, (c) $n = 4$ and (d) $n \to \infty$ (viz, the amoeba $\mathcal{A}_P$ itself).

5.2 Reproduction from Weights

Since the MLP has a rather high accuracy predicting the genus, we may use the weight matrices and bias vectors to obtain a purely linear expression for the genus. Again, let us illustrate this with $n = 1$. With the projected vector (x, y) as input, it turns out that the minimal network with such performance only requires one hidden layer with four neurons. This means that after passing the data to the MLP, we get the expression

$$p = W_2 \cdot \text{ReLU}(W_1 \cdot (x, y)^{\text{T}} + \boldsymbol{b}_1) + \boldsymbol{b}_2, \tag{5.3}$$

where $W_{1,2}$ are weight matrices and $\boldsymbol{b}_{1,2}$ are the bias vectors. For instance, in our training, we obtain

$$\begin{aligned} W_1 &= \begin{pmatrix} 0.4667 & 1.4986 & -0.3313 & -1.3031 \\ -1.3543 & -1.0408 & -0.5699 & -1.5610 \end{pmatrix}^{\text{T}}, \\ W_2 &= \begin{pmatrix} -1.1199 & 0.6904 & 0.6216 & 0.2827 \end{pmatrix} \end{aligned} \tag{5.4}$$

and

$$\boldsymbol{b}_1 = (-0.0349, 0.0439, 0.3044, -0.5884)^{\mathrm{T}},\ \boldsymbol{b}_2 = (-4.1003). \tag{5.5}$$

Then the genus is simply given by $g = \theta(p)$ where θ is the Heaviside function.

One may also analyze the network structures for higher n. For example, still with fewer than 2000 training samples, an MLP (with a slightly larger size) can give over 0.9 accuracy for $\mathcal{A}_P$, namely $n \to \infty$. The long expression determining the genus from the MLP was listed in [31], and we shall not repeat it here.

Likewise, we can consider lattice polygons with more internal points which would lead to a larger range of the genus. The manifold learnings and the approximated expressions from the neural network can be applied to these cases in a similar manner although the analysis would be more involved. More examples can be found in [31], along with CNN application to genus identification from Monte-Carlo generated images.

6 Brain Webs for Brane Webs

In Type IIB String Theory brane webs are configurations of (p, q) 5-branes, containing 5-brane line segments and junctions, where three or more 5-branes meet. We assume that the (p, q) 5-branes end on (p, q) 7-branes which are depicted as dots in the plane. The (p, q) charges of the 7-brane define an $SL(2, \mathbb{Z})$ monodromy action $M_{(p,q)}$ as one encircles it. We account for this by introducing a branch cut emanating from the 7-brane.

The data specifying a brane web can be encoded in a web matrix which lists the charges of the (p, q) 7-branes, multiplied by the number n of copies of each 5-brane.

$$W = \begin{pmatrix} n_1 p_1 & n_2 p_2 & \cdots & n_L p_L \\ n_1 q_1 & n_2 q_2 & \cdots & n_L q_L \end{pmatrix}, \tag{6.1}$$

where L is the number of external legs of the web[6].

A planar configuration of three or more 5-branes meeting at a point describes a 5d SCFT [71, 72]. Thus classifying 5-brane webs, corresponds to constructing 5d SCFTs. There are of course an infinite number of such webs but the space is hugely redundant, as infinitely many seemingly different W's actually correspond to physically equivalent configurations. This is due to two reasons: firstly the $\mathrm{SL}(2, \mathbb{Z})$ self-duality of Type IIB String Theory identifies congruent W's; secondly 7-branes can be moved along the (p, q) line past the intersection point in a Hanany-Witten move [73] without changing the low energy physics. Thus the classification of webs requires to mod out the space of webs by these equivalence relations.

The problem of classifying brane webs for $L = 3$ is in fact equivalent to the classification of sets of 7-branes. In [74] it was conjectured that inequivalent sets of 7-branes are characterised by the total monodromy of the webs, defined as the product of the monodromies of all the 7-branes

$$M_{\mathrm{tot}} = M_{(p_1,q_1)} M_{(p_2,q_2)} \cdots M_{(p_L,q_L)} \tag{6.2}$$

[6] We work with the convention that all the charges of the 5-branes are ingoing and the legs are ordered anticlockwise around the junction.

and the asymptotic charge invariant ℓ, defined as

$$\ell = \gcd\left\{\det\begin{pmatrix} p_i & p_j \\ q_i & q_j \end{pmatrix}, \forall\, i,j\right\}. \tag{6.3}$$

However, as counter examples in [32] explicitly show, these classifiers are not enough to fully specify a set of 7-branes, and are thus to be thought of as necessary but not sufficient conditions for equivalence. With this in mind, we define the following two notions of equivalent webs:

- **Strong equivalence:** Two webs are strongly equivalent if they can be transformed into each other by means of any combination of $\mathrm{SL}(2,\mathbb{Z})$ and Hanany-Witten moves.
- **Weak equivalence:** Two webs are weakly equivalent if they have the same rank, ℓ, and M_{tot} up to $\mathrm{SL}(2,\mathbb{Z})$.

The goal of [32], as reviewed here, was to teach a Siamese neural network (SNN) to identify equivalent webs, given by their web matrices (6.1), according to these two notions. Specifically an SNN was trained that took as input 2×3 matrices W_i and produced a 10-dimensional embedding $\mathbf{x}_i \in \mathbb{R}^{10}$ for each web matrix such that the Euclidean distance between embeddings of equivalent webs was minimised and the distanced between inequivalent webs maximised. Two webs were predicted to be equivalent if the squared Euclidean distance between their embeddings was less than some threshold value.

6.1 Weak Equivalence

Considering the simplest case of just two equal sized classes $\mathbf{X}_1$ and $\mathbf{X}_2$ of weakly equivalent webs (48 webs per class), the SNN was trained to determine whether a pair of webs from the set $\mathbf{X}_1 \cup \mathbf{X}_2$ belonged to the same group. It was observed that, after training on 80% of the data, the network determines weak equivalence with 100% accuracy on the remaining 20%. This result is visualised in Figure 6.1 by t-distributed stochastic neighbour embedding (t-SNE). The webs group together into two distinct weakly equivalent clusters.

Motivated by this result the investigation was extended to consider more classes, namely 14. For more than two equal sized classes there will be more inequivalent pairs of webs than equivalent and so to prevent an unbiased accuracy score the SNN was trained and tested on an equal number of equivalent and inequivalent pairs. The results dropped to a more modest 77% but the network still performed significantly better than random guessing.

6.2 Strong Equivalence

The same procedure was repeated using the notion of strong equivalence. With just two classes $\mathbf{Y}_1$ and $\mathbf{Y}_2$ (also 48 webs per class), the network predicted strong equivalence of pairs of webs from 20% of $\mathbf{Y}_1 \cup \mathbf{Y}_2$ with an accuracy of 50%, after training on 80%. This means that the network is no better than random guessing. It can be seen from the t-SNE plot in Figure 6.1 that the embeddings of $\mathbf{Y}_1$ and $\mathbf{Y}_2$ are mixed together and completely indistinguishable. Again the investigation was extended to 14 classes $\mathbf{Y}_I, I = 1, ..., 14$. Here the network predicted equivalence in the test set with accuracy of 50%.

(a) $\mathbf{X}_1 \cup \mathbf{X}_2$ (b) $\mathbf{Y}_1 \cup \mathbf{Y}_2$

Figure 6.1: t-SNE plots of the 10-dimensional web embeddings generated in the SNN of the webs from the respectively labelled datasets, reduced to 2 dimensions.

7 Quivers and Mutation

A rank n cluster algebra is a constructively defined subalgebra of the field of rational functions in n variables, with a set of generators (cluster variables) grouped into overlapping subsets (clusters) of size n, such that certain exchange relations allow one to transition from one cluster to another. These exchange relations, known as cluster mutations, can be described using the language of quivers.

Beyond representation theory, quivers physically provide a convenient representation of the particle content of gauge theories that describes D-branes probing varieties. A quiver diagram, Q, is a finite directed multigraph where each node represents a compact gauge factor $\mathrm{U}(N_i)$ of the total gauge group $\prod_i \mathrm{U}(N_i)$ and edges represent bi-fundamental and adjoint chiral multiplets.

All quivers we consider have skew-symmetric exchange matrices and thus given an initial cluster $\{x_1, \ldots, x_n\}$, we can define the exchange relation replacing cluster variable x_j by x'_j in the cluster as

$$x_j x'_j = \prod_{i \to j \text{ in } Q} x_i + \prod_{j \to k \text{ in } Q} x_k \tag{7.1}$$

for each $1 \leq j \leq n$, and where the products are over all incoming arrows and outgoing arrows, respectively. We thus get a new generator, x'_j, yielding a new cluster $\{x_1, \ldots, x_{j-1}, x'_j, x_{j+1}, \ldots, x_n\}$. Note that this is not the most general form of cluster mutation, nor quiver mutation as one may include updates to the gauge group rank information. The process of mutation also updates the quiver by the following:

1. Replace every incoming arrow $i \to j$ with the outgoing arrow $j \to i$ and replace every outgoing arrow $j \to k$ with the incoming arrow $k \to j$.

2. For every 2-path $i \to j \to k$ add a new arrow $i \to k$.

3. Remove all 2-cycles created by step 2.

Given a quiver Q, we construct the associated cluster algebra $\mathcal{A}_Q$ by applying cluster mutation in all directions and iterating to obtain the full list of cluster variables. Generically, this process yields an infinite number of generators, i.e. cluster variables, as well

as an infinite number of different quivers. Cluster algebras with a finite number of cluster variables are called finite type cluster algebras and cluster algebras of finite mutation type are those with a finite number of quivers.

Cluster algebras and quiver gauge theories were brought together in [75–77]. It was shown that the cluster algebra mutation rules can be realised as Seiberg dualities of quiver gauge theories, where Seiberg duality is a generalisation of the classical electro-magnetic duality for 4d $\mathcal{N} = 1$ supersymmetric gauge theories, this interpretation is the focus of the work in [33] of which we summarise here. This work is also extended from quiver mutation to cluster mutation in [78].

Our goal is to teach the machine to detect the mutation class of a given quiver. We consider two problems: the first is deciding whether two quivers are part of the same mutation class, and the second is determining specifically to which mutation class a given quiver belongs to. There are three types of quivers which we consider separately: finite type, finite mutation type which are not finite type (as finite mutation naturally include finite type), and infinite mutation type.

7.1 Gathering Quiver Data

The information of a quiver can be encoded in an adjacency matrix. It is these adjacency matrices together with the associated mutation class of the quiver that are fed to the machine. The list of quivers considered in the following investigations are shown in Figure 7.1 below. They are listed with an adjacency matrix representation and are labelled in the form **Qi**.

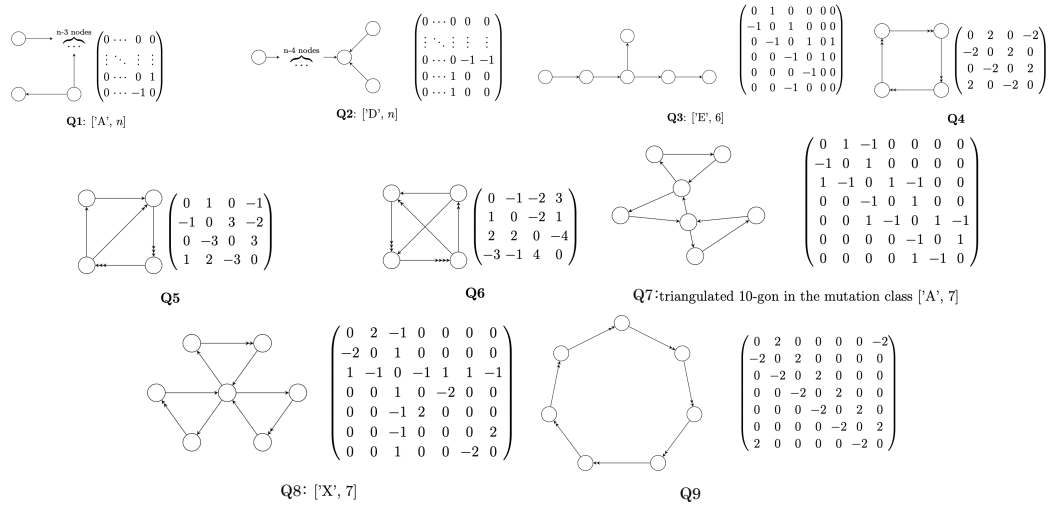


Figure 7.1: Quivers

7.2 Classifying Mutation Classes

We start off by considering the binary classification task where the input data consists of pairs of adjacency matrices M_1, M_2 that describe two quivers and the output is either 0 or 1, where 1 indicates that M_1 and M_2 are in the same Seiberg duality class and 0 indicates

that they are not. We apply the `Mathematica` built-in function `Classify` with Naïve Bayes (NB) method specified to carry out this classification.

On the two finite mutation type classes A4 and D4 (**Q1** and **Q2** in Figure 7.1), this gives 100% accuracy over 5-fold cross validation. For the case of two infinite type mutation classes, namely **Q4** and **Q5** in Figure 7.1, we also get 100% accuracy.

So far we have looked at one type at a time. We wonder whether different types would affect our results. A simple check would involve only two mutation classes with one Dynkin (i.e. finite type) and one affine (i.e. finite-mutation type but not finite). For instance, we test D4 and A(3,1)1 here. The `Classify` function again gives 100% accuracy as for the example of A4 and D4. This is an exciting result. We find that learning mutation classes of the same type (e.g. only Dynkin) and learning those of different types (e.g. Dynkin + affine) have the same performance. Let us further try an example with one finite type D4 and one infinite mutation type, namely **Q4**. We still get perfect accuracy. It appears that the mutation types do not really affect our learning performance for NB.

We now contemplate datasets containing more mutation classes. For the finite type we use A6, D6 and E6, corresponding to **Q1**, **Q2**, and **Q3** respectively. The learning result of 5-fold validation is 90% which, albeit is not as perfect as the case with two classes, is still very good. For the infinite mutation type we add **Q6** to **Q4** and **Q5** and we also see a drop from 100% to 90% accuracy. We see that the numbers of different classes does affect the performance of NB in determining whether two quivers are of the same mutation class.

7.3 Multiclass Classification

Besides pairing matrices and assigning 1's and 0's, there is a more direct way to classify theories, we can simply assign different mutation classes with different labels and then let the machine tell which classes the given quivers belong to. We turn to `Python` to perform this task using a CNN with the help of `Sage` [79, 80] and `Tensorflow`.

We choose three classes, namely **Q7**, **Q8** and **Q9** in Figure 7.1, the first two classes are finite while the third is infinite. We label the three classes $[1,0,0]$, $[0,1,0]$ and $[0,0,1]$ respectively. Thus when the machine predicts $[a_1, a_2, a_3]$, it is giving probabilities of the three classes respectively. We find that there is only $\sim 55\%$ accuracy when the machine is trained on 80% of the data. However, remarkably on the last class, which is infinite, the machine has 100% accuracy, i.e. it always correctly recognises the matrices in this class and never misclassifies other matrices to this class. Hence the machine seems to have learnt something related to finite and infinite mutations.

8 Dessins d'Enfants

Dessins d'Enfants, or children's drawings, are bipartite graphs drawn on Riemann surfaces that graphically represent the degeneracy information of Belyĭ maps from the surface to the Riemann sphere. Where these Belyĭ maps exist, via Belyĭ's theorem, one can define the Riemann surface using exclusively algebraic numbers $\overline{\mathbb{Q}}$ [81].

These dessins were hypothesised by Grothendieck to faithfully represent the absolute Galois group over the rationals, a famously elusive object [82]. Although the dessins

are easy to draw, the computation of the underlying Belyĭ map is notoriously difficult; and hence extracting any information about this map from the dessin alone is especially advantageous for uncovering the mysteries behind their connections to Galois groups.

It should be noted that dessins have an array of applications in physics also. They may be considered an interpretation of dimer models [83–86], arise naturally when working with K3 surfaces [87, 88] which may act as the moduli spaces for specific quiver gauge theories, and have been useful in factorising Seiberg-Witten curves [89, 90].

In this section we review the work of [34], which seeks to apply supervised machine learning techniques to this programme of recovering Belyĭ map information from the respective dessins. The focus is on extracting the size of the Galois orbits that the dessins belong to from tensor representations of the graphs. The dataset of dessins used arises from elliptic K3 surfaces, where surprisingly their j-invariants are Belyĭ in nature and hence produce dessins.

8.1 Defining the Data

Belyĭ's theorem states that a Riemann surface, X, can be defined over algebraic numbers if and only if there exists a map to the Riemann sphere, $\beta : X \longmapsto \mathbb{P}^1_{\mathbb{C}}$, which has exactly three ramification points. Due to the symmetry under the Möbius group of the Riemann sphere, these three ramification points can be mapped to $\{0, 1, \infty\}$. Then the preimages for the interval [0,1] on the Riemann sphere through the Belyĭ map, β, can be calculated.

Each of these preimages on the Riemann sphere can be affiliated to the bipartite structure of the dessin: $\beta^{-1}(0) \mapsto \circ$, $\beta^{-1}(1) \mapsto \bullet$, $\beta^{-1}(0,1) \mapsto -$. Within this structure a vertex's degree indicates the point's ramification index, i.e. the Taylor expansion of β about that preimage point beyond the constant term will begin at some order $n > 1$, being the ramification index. Further to this each face is associated to a preimage of ∞, and half the number of bounding edges to the face indicates that point's ramification index.

As an example consider the Belyĭ map

$$\beta(z) = \frac{(z^8 - 14z^4 + 1)^3}{(-108z^4(z^4 + 1)^4)}\,, \tag{8.1}$$

which has ramification information $\{3^8|2^{12}|4^6\}$ and is drawn in Figure 8.1a. The ramification information indicates the shown 8 white vertices of valence 3, 12 black vertices of valence 2, and 6 faces with 8 bounding edges (including the external face as truly drawn on the Riemann sphere).

As can be seen in this example, β is defined over $\mathbb{Q}$. However in other examples the map may require non-rational numbers. Each of these non-rational numbers in the map definition amount to a field extension of $\mathbb{Q}$ towards $\overline{\mathbb{Q}}$. The minimal polynomial over $\mathbb{Q}$ with each non-rational number as a root will have other roots also, and changing the root occurring in the map to all the other roots amounts to taking this Belyĭ map, and respective dessin which changes each time, through its Galois orbit. For the example above as no non-rational numbers occur the Galois orbit is size 1, and in the database considered they vary up to size 4.

(a) (b)

Figure 8.1: An example dessin d'enfant (a), of the type examined in this work, it is a planar dessin, and also clean - as all same colour nodes have the same valency. Then the same example dessin with edges labelled (b) for the cyclic edge list representation: {{{8,1,9},{2,3,10},{4,5,11},{12,6,7},{18,14,19},{22,16,23},{13,24,17},{21,20,15}}, {{1,2},{9,13},{10,14},{3,4},{15,11},{24,23},{17,18},{8,7},{5,6},{12,16},{21,22},{20,19}}}.

The dataset machine learnt comes from [88]. These 191 dessins come from the modular group $\mathrm{PSL}(2,\mathbb{Z})$, specifically each of the 112 index 24 torsion-free genus-zero subgroups. Performing quotient action by each of these subgroups on the upper half-plane fibred with a complex line produces an elliptic K3 surface whose j-invariant is surprisingly always a Belyĭ map! Then taking each of these maps through their Galois orbits produces the 191 dessins considered.

8.2 Learning Galois Orbit Size

These 191 dessins each have a respective Galois orbit size from $\{1,2,3,4\}$. The machine learning approach was hence designed to apply a supervised neural network classifier to learning this Galois orbit size from an input dessin represented tensorially.

The initial method of representation ignored the bivalent black nodes, absorbing them into the edges, and represented each dessin by its adjacency matrix on the white nodes, $\mathcal{M}_{8\mathrm{x}8}$. However, even with data augmentation to increase the data size, the learning was not strong with performance only marginally better than random guessing. This poor performance was attributed to the adjacency matrix representation not being faithful, i.e. there were occasional scenarios where different dessins had the same adjacency matrix.

To design a faithful representation method, the category equivalence between dessins and finite sets under permutation was used. This time all the edges were labelled, and at each white node the edges were listed clockwise, and at each black node anticlockwise (although direction redundant here as they are bivalent). This representation method is shown in Figure 8.1b, and admitted its own methods for data augmentation of relabelling edges, cyclically shuffling the edge lists, or shuffling the node lists. The edge lists were then flattened to give the final input vectors passed as input to the NNs, $\mathcal{V}_{48}$.

For the adjacency matrix format, augmentation of the 152 distinct dessin matrices with ~ 1000 permutations of rows/columns (excluding those which didn't change the matrix) lead to a proportional distribution over the orbit size classes of: $[0.49, 0.31, 0.19, 0.01]$. Conversely, the equivalent data distribution after augmenting the edge list representation by 1000 permutations was: $[0.40, 0.35, 0.23, 0.02]$.

Data Format	Performance Measure	
	Accuracy	MCC
$\mathcal{M}_{8\text{x}8}$	0.536 $\pm$0.002	0.180 $\pm$0.008
$\mathcal{V}_{48}$	0.920 $\pm$0.003	0.880 $\pm$0.040

Table 8.1: Accuracy and MCC learning results for the investigations, constituting 2 primary data formats: the adjacency matrix format $\mathcal{M}_{8\text{x}8}$ and the cyclic edge list format $\mathcal{V}_{48}$.

The NN classifiers of 4 layers of 512 neurons with LeakyReLU activation and 0.25 dropout were trained using an Adam optimiser over the cross-entropy loss. Due to the bias of the data sets towards lower orbit size classes, as well as the prototypical accuracy metric MCC was also used and provides a better measure of the learning performance. In addition 5-fold cross-validation was performed for each investigation such that the measures could be averaged and standard error calculated to provide confidence in the results.

Whilst the adjacency matrix results were only marginally better than random guessing (exemplified by an MCC close to 0), the results using the cyclic edge list representation were far better, as shown in Table 8.1. The performance indicates that the Galois orbit size is perhaps combinatorically encoded in the dessin information, in such a way that this simple ML architecture can take advantage of it.

9 Hessian Manifolds, Optimal Transport and GANs

In [91–94], the optimal transport problem and Monge-Ampère equations were related to Kähler geometry. In particular, consider a compact toric manifold $\mathcal{M}$ with Kähler metric invariant under T^n in the $(\mathbb{C}^*)^n$-action. Then the study of this complex manifold could mostly be reduced to the convex analysis on $\mathbb{R}^n$. A plurisubharmonic function ψ invariant under T^n can be represented by some convex function ϕ on $\mathbb{R}^n$ via[7] [95]

$$\text{Log}_* c_n(\partial\bar{\partial}\psi)^n = \det\left(\frac{\partial^2\phi}{\partial x_i \partial x_j}\right) \mathrm{d}V, \tag{9.1}$$

where x_i's are the coordinates on $\mathbb{R}^n$ and $\mathrm{d}V$ denotes the Euclidean volume form on $\mathbb{C}^n$. Here, the map

$$\text{Log}: \ (\mathbb{C}^*)^n \to \mathbb{R}^n, \ z \mapsto \log(z) =: x \tag{9.2}$$

is defined such that the j^{th} component of x satisfies $x^j = \log\left(|z^j|^2\right)$. By an abuse of notation, we abbreviate the above equation as

$$c_n(\partial\bar{\partial}\psi)^n = \det\left(\frac{\partial^2\phi}{\partial x_i \partial x_j}\right) \mathrm{d}V. \tag{9.3}$$

[7] There could be different conventions for the constant c_n, such as $c_n = \frac{i^n}{(2\pi)^n n!}$ in [95]. However, this is not really important here as it could be absorbed into ψ or ϕ.

Therefore, the *complex* Monge-Ampère operator in this case can be represented by the Hessian determinant of the real ϕ, or in other words, reduced to the *real* Monge-Ampère equation

$$\det\left(\frac{\partial^2\phi}{\partial x_i\partial x_j}\right) = \mathrm{e}^{-\lambda\phi} \tag{9.4}$$

for $\lambda \in \mathbb{R}$. In fact, this endows $\mathcal{M}$ with a Hessian metric which reads

$$g = \nabla \mathrm{d}\phi_k = \frac{\partial^2\phi_k}{\partial x_i\partial x_j}\mathrm{d}x_i \otimes \mathrm{d}x_j \tag{9.5}$$

for some smooth $\phi_k : U_k \to \mathbb{R}$ in terms of the local covering $\{U_k\}$ in the charts. The right hand side of (9.3) is known as the real Monge-Ampère measure $\mathrm{MA}(\phi)$ of the Hessian metric, where ϕ is the shorthand notation for $\{\phi_k\}$. Moreover, we require $\mathcal{M}$ to be special[8] such that $\mathrm{MA}(\phi)$ is invariant under coordinate transformations and hence globally defined on $\mathcal{M}$. The existence and uniqueness of such $\mathrm{MA}(\phi)$ was proven in [96].

9.1 Optimal Transport

The aim of the optimal transport problem is to minimize some cost function $c : X \times Y \to \mathbb{R}$, for some probability spaces X and Y. In the setting of Hessian manifolds, $X = \mathcal{M}$ and $Y = \mathcal{M}^*$, where $\mathcal{M}^*$ is the dual Hessian manifold [92]. We shall not expound the details of dual Hessian manifolds here, and it suffices to know that $\mathrm{d}\phi : \mathcal{M} \to \mathcal{M}^*$ is a diffeomorphism. Given the Borel probability measures μ and ν on $\mathcal{M}$ and $\mathcal{M}^*$ respectively, we can define

Definition 9.1. *The* ν-Monge-Ampère measure *of a convex section* ϕ *is* $\mathrm{MA}_\nu(\phi) = (T_\phi)_*\nu$. *Here* T_ϕ*, which can be identified as* $\mathrm{d}(\phi^*)$*, is the inverse of* $\mathrm{d}\phi$ *(defined almost everywhere on* $\mathcal{M}^*$*).*

We now state the optimal transport problem and the associated cost function.

Definition 9.2. *Let* $(\mathcal{M}, L)$ *be a compact Hessian manifold with convex section* ϕ_0 *and* Π *the fundamental group of* $\mathcal{M}$*. The cost function* $c : \mathcal{M} \times \mathcal{M}^* \to \mathbb{R}$ *is* $c(x,y) = -[x,y] + \phi_0(x) + \phi_0^*(y)$*. The pairing operation is defined as* $[x,y] := \sup_{\gamma\in\Pi} \gamma \cdot q(x) - q$ *for* q *being a point in the fibre of* K^* *over* y*, and* $\phi_0^*(y) = [x,y] - \phi_0(x)$ *is the Legendre transform*[9]*. Then the optimal transport problem is to minimize the transportation cost* $I_c = \int_{\mathcal{M}\times\mathcal{M}^*} c(x,y)\mathrm{d}\rho$ *for all probability meausures* ρ *on* $\mathcal{M} \times \mathcal{M}^*$*.*

Let ϕ be in the Kähler class of ϕ_0. Consider the cost function $c'(x,y)$ induced by ϕ. Then $c'(x,y) = c(x,y) - f - f^c$, where $f := (\phi - \phi_0)$ and $f^c := (\phi^* - \phi_0^*)$. In terms of the Kantorovich problem in optimal transport, this means that c' is equivalent to c. In other

[8] An affine manifold is said to be special if $\mathrm{d}V$ is preserved by the transition maps, viz, $\mathrm{Hol}(\nabla) \subset \mathrm{SL}(n,\mathbb{R})$.

[9] The definition of [-,-] is independent of the choice of q. Hence this is well-defined.

words, c' minimizes I_c as well since

$$\begin{aligned} I_{c'} &= \int_{\mathcal{M}\times\mathcal{M}^*} c'(x,y)\mathrm{d}\rho = \int_{\mathcal{M}\times\mathcal{M}^*} c(x,y)\mathrm{d}\rho - \int_{\mathcal{M}} f\mathrm{d}\rho - \int_{\mathcal{M}^*} f^c\mathrm{d}\rho \\ &= \int_{\mathcal{M}\times\mathcal{M}^*} c(x,y)\mathrm{d}\rho - \int_{\mathcal{M}} f\mathrm{d}\mu - \int_{\mathcal{M}^*} f^c\mathrm{d}\nu = I_c + C, \end{aligned} \tag{9.6}$$

where C is a constant independent of ρ.

It was shown in [92] that the cost function is the squared distance induced by the flat Riemannian metric d determined by L, viz, $c(x,y) = \frac{d^2(x,y)}{2}$. From above, we can also write the dual Kantorovich problem, that is, to maximize the quantity $J = \int_{\mathcal{M}} f\mathrm{d}\mu + \int_{\mathcal{M}^*} f^c\mathrm{d}\nu$, where f is known as the Kantorovich potential and f^c is called its c-transform.

Now we can relate the Monge-Ampère measures to the optimal transport problem [92].

Theorem 9.1. *There exists a smooth, strictly convex section ϕ such that $\mathrm{MA}_\nu(\phi) = (T_\phi)_*\nu = \mu$. Then $\mathrm{d}\phi : \mathcal{M} \to \mathcal{M}^*$ is the optimal transport map determined by μ, ν and $c(x,y)$. In other words, such ϕ minimizes the transportation cost I_c.*

Here, we discussed how special Hessian manifolds can be studied via optimal transport. It would also be interesting to study the relations of real Monge-Ampère equations on real tori and complex Monge-Ampère equations on Abelian manifolds as in [91]. In terms of mirror symmetry and tropical geometry, solutions to real Monge-Ampère equations on Hessian manifolds appear as large complex limits of Kähler-Einstein metrics on complex mainfolds [97]. It would also be interesting to extend the study to singular or non-compact settings in the context of mirror symmetry.

9.2 GANs

A generative adversarial network (GAN) [98] is composed of two neural networks known as the generator and the discriminator. Given the training data, the generator captures its distribution from the latent space and generates new samples. On the other hand, the discriminator estimates the probability of the samples coming from the true data or from the generator. The two networks are trained simultaneously, and the competition between them improves the performance of the machine until the generated samples cannot be distinguished from the true data.

In [99], the improvement of the GAN model was attempted using the theory of optimal transport. Let X be the image space as a manifold embedded in $\mathbb{R}^n$. The latent space is then some $\mathbb{R}^m$ which is related to X under the usual homeomorphism in the charts $\psi_i : U_i \to \mathbb{R}^m$, and hence ψ_i corresponds to the decoder NN. Then the generative network and the discriminative network both have certain data distributions, say, μ and ν. The goal of GANs is therefore to minimize the Wasserstein cost $W_c = \int_{X\times X} c(x,y)\mathrm{d}\rho$ for some cost function $c : X \times X \to \mathbb{R}$, where ρ is the Borel probability measures on $X \times X$. Now that this becomes an optimal transport problem, it is natural to have a correspondence between the geometry side and NN side.

9.3 The Dictionary

Since $\mathcal{M}$ and $\mathcal{M}^*$ are equivalent as affine manifolds, both of them correspond to the image space X inside the ambient space. It is also natural to match the corresponding probability measures $\mu_{\text{hess}} \leftrightarrow \mu_{\text{NN}}$ and $\nu_{\text{hess}} \leftrightarrow \nu_{\text{NN}}$.

We can then equate the transportation costs $I_c \leftrightarrow W_c$ as well. In other words, the two cost functions should also match. Indeed, in the language of [99], we have $c_{\text{NN}}(\boldsymbol{x}, \boldsymbol{y}) = |\boldsymbol{x} - \boldsymbol{y}|^2/2$. As argued in [99], the discriminator function, denoted by ϕ_ξ, which is the metric between distributions in the discriminator, is equivalent to the Kantorovich potential. Hence, for the dual Kantorovich problem, we have $f \leftrightarrow \phi_\xi$ as well as $f^c = c_{\text{hess}}(x, y) - f \leftrightarrow \phi_\xi^c = c_{\text{NN}}(x, y) - \phi_\xi$.

In light of [99], the correpsonding Monge-Ampère equation for GAN is

$$\det\left(\frac{\partial^2 u}{\partial x_i \partial x_j}\right) = \frac{\mu(x)}{\nu(\nabla u(x))}, \tag{9.7}$$

where u is called the Brenier potential, and ∇u is the transportation map that optimizes our problem. In the GAN model, u is often represented by linear combinations and ReLUs. Then we can write the correspondence for the two Brenier problems via $\phi \leftrightarrow u$ and $\mathrm{d}\phi \leftrightarrow \nabla u$. Recall that T_ϕ is the inverse of $\mathrm{d}\phi$. Thus, the Brenier problem formulated by $(\nabla u)_\sharp \mu = \nu$ in [99] should correpsond to $(T_\phi)_* \nu = \mu$ in the dictionary. Indeed, solving these two equations gives the optimal transport maps of the two problems.

Since the Kähler potential, say ψ, is plush, in terms of (9.3), we could write

$$c_n (\partial\bar{\partial}\psi)^n = \frac{\mu(x)}{\nu(\nabla u(x))} \mathrm{d}V, \tag{9.8}$$

whose left hand side could be expanded as

$$\text{LHS} = c_n \epsilon^{\mu_1 \cdots \mu_n \nu_1 \cdots \nu_n} \prod_{i,j} \frac{\partial^2 \psi}{\partial z^{\mu_i} \partial \bar{z}^{\nu_j}} \mathrm{d}z^1 \wedge \cdots \wedge \mathrm{d}z^n \wedge \mathrm{d}\bar{z}^1 \wedge \cdots \wedge \mathrm{d}\bar{z}^n = c_n \det\left(\frac{\partial^2 \psi}{\partial z^\mu \partial \bar{z}^\nu}\right) \mathrm{d}V. \tag{9.9}$$

Therefore, we may write[10]

$$c_n \det\left(\frac{\partial^2 \psi}{\partial z^\mu \partial \bar{z}^\nu}\right) = \frac{\mu(x)}{\nu(\nabla u(x))}, \tag{9.10}$$

and we could relate the Kähler potential on the geometry side to the Brenier potential on the NN side.

Here, we identify certain manifolds in Kähler geometry with certain models in NNs by viewing both of them from the perspective of optimal transport. When defining the dual Hessian manifold, one needs to introduce the so-called affine $\mathbb{R}$-bundle over $\mathcal{M}$. We have not provided anything on the NN side that corresponds to this bundle structure. In [100], an NN learning system is regarded as a fibre bundle. It might be possible that we could find some correspondence there.

[10] Recall that there is a Log_* hidden in the front of the left hand side.

Acknowledgements

The authors wish to thank their numerous collaborators on these projects. JB is supported by a CSC scholarship. 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 the PhD studentship.

References

[1] Y.-H. He, "Deep-Learning the Landscape," arXiv:1706.02714 [hep-th].

[2] J. Carifio, J. Halverson, D. Krioukov, and B. D. Nelson, "Machine Learning in the String Landscape," *JHEP* **09** (2017) 157, arXiv:1707.00655 [hep-th].

[3] D. Krefl and R.-K. Seong, "Machine Learning of Calabi-Yau Volumes," *Phys. Rev. D* **96** no. 6, (2017) 066014, arXiv:1706.03346 [hep-th].

[4] F. Ruehle, "Evolving neural networks with genetic algorithms to study the String Landscape," *JHEP* **08** (2017) 038, arXiv:1706.07024 [hep-th].

[5] Y.-H. He, "Machine-learning the string landscape," *Phys. Lett. B* **774** (2017) 564–568.

[6] M. R. Douglas, R. L. Karp, S. Lukic, and R. Reinbacher, "Numerical Calabi-Yau metrics," *J. Math. Phys.* **49** (2008) 032302, arXiv:hep-th/0612075.

[7] K. Bull, Y.-H. He, V. Jejjala, and C. Mishra, "Machine Learning CICY Threefolds," *Phys. Lett. B* **785** (2018) 65–72, arXiv:1806.03121 [hep-th].

[8] K. Bull, Y.-H. He, V. Jejjala, and C. Mishra, "Getting CICY High," *Phys. Lett. B* **795** (2019) 700–706, arXiv:1903.03113 [hep-th].

[9] Y.-H. He and A. Lukas, "Machine Learning Calabi-Yau Four-folds," *Phys. Lett. B* **815** (2021) 136139, arXiv:2009.02544 [hep-th].

[10] L. B. Anderson, M. Gerdes, J. Gray, S. Krippendorf, N. Raghuram, and F. Ruehle, "Moduli-dependent Calabi-Yau and SU(3)-structure metrics from Machine Learning," *JHEP* **05** (2021) 013, arXiv:2012.04656 [hep-th].

[11] V. Jejjala, D. K. Mayorga Pena, and C. Mishra, "Neural Network Approximations for Calabi-Yau Metrics," arXiv:2012.15821 [hep-th].

[12] M. R. Douglas, S. Lakshminarasimhan, and Y. Qi, "Numerical Calabi-Yau metrics from holomorphic networks," arXiv:2012.04797 [hep-th].

[13] H. Erbin, R. Finotello, R. Schneider, and M. Tamaazousti, "Deep multi-task mining Calabi–Yau four-folds," *Mach. Learn. Sci. Tech.* **3** no. 1, (2022) 015006, arXiv:2108.02221 [hep-th].

[14] M. Larfors, A. Lukas, F. Ruehle, and R. Schneider, "Learning Size and Shape of Calabi-Yau Spaces," arXiv:2111.01436 [hep-th].

[15] Y.-H. He, S. Lal, and M. Z. Zaz, "The World in a Grain of Sand: Condensing the String Vacuum Degeneracy," arXiv:2111.04761 [hep-th].

[16] S. Abel, A. Constantin, T. R. Harvey, and A. Lukas, "Evolving Heterotic Gauge Backgrounds: Genetic Algorithms versus Reinforcement Learning," arXiv:2110.14029 [hep-th].

[17] A. Constantin, T. R. Harvey, and A. Lukas, "Heterotic String Model Building with Monad Bundles and Reinforcement Learning," arXiv:2108.07316 [hep-th].

[18] A. Ashmore, R. Deen, Y.-H. He, and B. A. Ovrut, "Machine learning line bundle connections," *Phys. Lett. B* **827** (2022) 136972, arXiv:2110.12483 [hep-th].

[19] X. Gao and H. Zou, "Applying machine learning to the Calabi-Yau orientifolds with string vacua," *Phys. Rev. D* **105** no. 4, (2022) 046017, arXiv:2112.04950 [hep-th].

[20] 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*. 11, 2021. arXiv:2111.11466 [hep-th].

[21] Y.-H. He, *The Calabi–Yau Landscape: From Geometry, to Physics, to Machine Learning*. Lecture Notes in Mathematics. 5, 2021. arXiv:1812.02893 [hep-th].

[22] Y.-H. He and M. Kim, "Learning Algebraic Structures: Preliminary Investigations," arXiv:1905.02263 [cs.LG].

[23] Y.-H. He and S.-T. Yau, "Graph Laplacians, Riemannian Manifolds and their Machine-Learning," arXiv:2006.16619 [math.CO].

[24] K. Buzzard, "Proving theorems with computers," *Notices of the American Mathematical Society* **67** no. 11, (2020) 1.

[25] Y.-H. He, "Machine-Learning Mathematical Structures," arXiv:2101.06317 [cs.LG].

[26] A. Davies, P. Veličković, L. Buesing, S. Blackwell, D. Zheng, N. Tomašev, R. Tanburn, P. Battaglia, C. Blundell, A. Juhász, *et al.*, "Advancing mathematics by guiding human intuition with ai," *Nature* **600** no. 7887, (2021) 70–74.

[27] Y.-H. He, "From the String Landscape to the Mathematical Landscape: a Machine-Learning Outlook," in *14th International Workshop on Lie Theory and Its Applications in Physics*. 2, 2022. arXiv:2202.06086 [hep-th].

[28] D. S. Berman, Y.-H. He, and E. Hirst, "Machine learning Calabi-Yau hypersurfaces," *Phys. Rev. D* **105** no. 6, (2022) 066002, arXiv:2112.06350 [hep-th].

[29] J. Bao, Y.-H. He, E. Hirst, J. Hofscheier, A. Kasprzyk, and S. Majumder, "Polytopes and Machine Learning," arXiv:2109.09602 [math.CO].

[30] J. Bao, Y.-H. He, E. Hirst, J. Hofscheier, A. Kasprzyk, and S. Majumder, "Hilbert series, machine learning, and applications to physics," *Phys. Lett. B* **827** (2022) 136966, arXiv:2103.13436 [hep-th].

[31] J. Bao, Y.-H. He, and E. Hirst, "Neurons on Amoebae," arXiv:2106.03695 [math.AG].

[32] G. Arias-Tamargo, Y.-H. He, E. Heyes, E. Hirst, and D. Rodriguez-Gomez, "Brain Webs for Brane Webs," arXiv:2202.05845 [hep-th].

[33] 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** no. 8, (2020) 086013, arXiv:2006.10783 [hep-th].

[34] Y.-H. He, E. Hirst, and T. Peterken, "Machine-learning dessins d'enfants: explorations via modular and Seiberg–Witten curves," *J. Phys. A* **54** no. 7, (2021) 075401, arXiv:2004.05218 [hep-th].

[35] P. Candelas, G. T. Horowitz, A. Strominger, and E. Witten, "Vacuum Configurations for Superstrings," *Nucl. Phys. B* **258** (1985) 46–74.

[36] J. Bao, Y.-H. He, E. Hirst, and S. Pietromonaco, "Lectures on the Calabi-Yau Landscape," arXiv:2001.01212 [hep-th].

[37] Y.-H. He, "Calabi-Yau Spaces in the String Landscape," arXiv:2006.16623 [hep-th].

[38] Y.-H. He, "Universes as big data," *Int. J. Mod. Phys. A* **36** no. 29, (2021) 2130017, arXiv:2011.14442 [hep-th].

[39] M. Abadi *et al.*, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015. https://www.tensorflow.org/. Software available from tensorflow.org.

[40] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research* **12** (2011) 2825–2830.

[41] B. Bengfort, R. Bilbro, N. Danielsen, L. Gray, K. McIntyre, P. Roman, Z. Poh, *et al.*, "Yellowbrick," 2018. http://www.scikit-yb.org/en/latest/.

[42] C. Tralie, N. Saul, and R. Bar-On, "Ripser.py: A lean persistent homology library for python," *The Journal of Open Source Software* **3** no. 29, (Sep, 2018) 925. https://doi.org/10.21105/joss.00925.

[43] P. Candelas, M. Lynker, and R. Schimmrigk, "Calabi-yau manifolds in weighted p4," *Nuclear Physics B* **341** no. 2, (1990) 383–402. https://www.sciencedirect.com/science/article/pii/055032139090185G.

[44] M. Kreuzer and H. Skarke, "Complete classification of reflexive polyhedra in four-dimensions," *Adv. Theor. Math. Phys.* **4** (2002) 1209–1230, arXiv:hep-th/0002240.

[45] C. Vafa, "String Vacua and Orbifoldized L-G Models," *Mod. Phys. Lett. A* **4** (1989) 1169.

[46] V. V. Batyrev, "On the stringy Hodge numbers of mirrors of quasi-smooth Calabi-Yau hypersurfaces," arXiv:2006.15825 [math.AG].

[47] M. Cirafici, "Persistent Homology and String Vacua," *JHEP* **03** (2016) 045, arXiv:1512.01170 [hep-th].

[48] A. Cole and G. Shiu, "Topological Data Analysis for the String Landscape," *JHEP* **03** (2019) 054, arXiv:1812.06960 [hep-th].

[49] A. M. Kasprzyk, "Canonical toric fano threefolds," *Canadian Journal of Mathematics* **62** no. 6, (2010) 1293–1309, arXiv:0806.2604 [math.AG].

[50] B. Bengfort and R. Bilbro, "Yellowbrick: Visualizing the Scikit-Learn Model Selection Process,". http://joss.theoj.org/papers/10.21105/joss.01075.

[51] I. Borg and P. J. Groenen, *Modern multidimensional scaling: Theory and applications*. Springer Science & Business Media, 2005.

[52] B. Feng, A. Hanany, and Y.-H. He, "Counting gauge invariants: The Plethystic program," *JHEP* **03** (2007) 090, arXiv:hep-th/0701063.

[53] D. Mehta, Y.-H. He, and J. D. Hauenstein, "Numerical Algebraic Geometry: A New Perspective on String and Gauge Theories," *JHEP* **07** (2012) 018, arXiv:1203.4235 [hep-th].

[54] I. Dolgachev, "Weighted projective varieties," in *Group actions and vector fields (Vancouver, B.C., 1981)*, vol. 956 of *Lecture Notes in Math.*, pp. 34–71. Springer, Berlin, 1982.

[55] M. F. Atiyah and I. G. Macdonald, *Introduction to commutative algebra*. Addison-Wesley Publishing Co., Reading, Mass.-London-Don Mills, Ont., 1969.

[56] R. P. Stanley, "Hilbert functions of graded algebras," *Advances in Mathematics* **28** (1978) 57–83.

[57] E. Hirst, "Machine Learning for Hilbert Series," 3, 2022. arXiv:2203.06073 [hep-th].

[58] G. Brown and A. M. Kasprzyk, "The Graded Ring Database." Online. http://www.grdb.co.uk/.

[59] G. Brown and A. M. Kasprzyk, "The Fano 3-fold database." *Zenodo* https://doi.org/10.5281/zenodo.5820338, 2022.

[60] S. Altınok, G. Brown, and M. Reid, "Fano 3-folds, $K3$ surfaces and graded rings," in *Topology and geometry: commemorating SISTAG*, vol. 314 of *Contemp. Math.*, pp. 25–53. Amer. Math. Soc., Providence, RI, 2002.

[61] G. Brown and A. M. Kasprzyk, "Kawamata boundedness for Fano threefolds and the Graded Ring Database." arXiv:2201.07178 [math.AG], 2022.

[62] G. Mikhalkin, "Amoebas of algebraic varieties and tropical geometry," *Different faces of geometry* (2004) 257–300, arXiv:math/0403015.

[63] I. Gelfand, M. Kapranov, and A. Zelevinsky, *Discriminants, Resultants, and Multidimensional Determinants*. Modern Birkhäuser Classics. Birkhäuser Boston, 2008.

[64] I. Itenberg, G. Mikhalkin, and E. I. Shustin, *Tropical algebraic geometry*, vol. 35. Springer Science & Business Media, 2009.

[65] R. Kenyon, A. Okounkov, and S. Sheffield, "Dimers and amoebae," arXiv:math-ph/0311005.

[66] B. Feng, Y.-H. He, K. D. Kennaway, and C. Vafa, "Dimer models from mirror symmetry and quivering amoebae," *Adv. Theor. Math. Phys.* **12** no. 3, (2008) 489–545, arXiv:hep-th/0511287.

[67] H. Ooguri and M. Yamazaki, "Crystal Melting and Toric Calabi-Yau Manifolds," *Commun. Math. Phys.* **292** (2009) 179–199, arXiv:0811.2801 [hep-th].

[68] A. Zahabi, "Quiver asymptotics and amoeba: Instantons on toric Calabi-Yau divisors," *Phys. Rev. D* **103** no. 8, (2021) 086024, arXiv:2006.14041 [hep-th].

[69] J. Bao, Y.-H. He, and A. Zahabi, "Mahler Measure for a Quiver Symphony," arXiv:2108.13903 [hep-th].

[70] K. Purbhoo, "A nullstellensatz for amoebas," *Duke Mathematical Journal* **141** no. 3, (2008) 407–445, arXiv:math/0603201.

[71] O. Aharony and A. Hanany, "Branes, superpotentials and superconformal fixed points," *Nucl. Phys. B* **504** (1997) 239–271, arXiv:hep-th/9704170.

[72] O. Aharony, A. Hanany, and B. Kol, "Webs of (p,q) five-branes, five-dimensional field theories and grid diagrams," *JHEP* **01** (1998) 002, arXiv:hep-th/9710116.

[73] A. Hanany and E. Witten, "Type IIB superstrings, BPS monopoles, and three-dimensional gauge dynamics," *Nucl. Phys. B* **492** (1997) 152–190, arXiv:hep-th/9611230.

[74] O. DeWolfe, T. Hauer, A. Iqbal, and B. Zwiebach, "Uncovering the symmetries on [p,q] seven-branes: Beyond the Kodaira classification," *Adv. Theor. Math. Phys.* **3** (1999) 1785–1833, arXiv:hep-th/9812028.

[75] B. Feng, A. Hanany, and Y.-H. He, "D-brane gauge theories from toric singularities and toric duality," *Nuclear Physics B* **595** no. 1-2, (Feb, 2001) 165–200. https://doi.org/10.1016%2Fs0550-3213%2800%2900699-4.

[76] B. Feng, A. Hanany, Y.-H. He, and A. M. Uranga, "Toric duality as seiberg duality and brane diamonds," *Journal of High Energy Physics* **2001** no. 12, (Dec, 2001) 035–035. https://doi.org/10.1088%2F1126-6708%2F2001%2F12%2F035.

[77] F. Cachazo, B. Fiol, K. Intriligator, S. Katz, and C. Vafa, "A geometric unification of dualities," *Nuclear Physics B* **628** no. 1-2, (Apr, 2002) 3–78. https://doi.org/10.1016%2Fs0550-3213%2802%2900078-0.

[78] P.-P. Dechant, Y.-H. He, E. Heyes, and E. Hirst, "Cluster Algebras: Network Science and Machine Learning," arXiv:2203.13847 [math.CO].

[79] W. Stein *et al.*, *Sage Mathematics Software (Version 9.4.0)*. The Sage Development Team, 2021. http://www.sagemath.org.

[80] G. Musiker and C. Stump, "A compendium on the cluster algebra and quiver package in sage," 2011.

[81] G. V. Belyĭ, "On Galois Extensions of a Maximal Cyclotomic Field," *Izvestiya: Mathematics* **14** no. 2, (Apr., 1980) 247–256.

[82] A. Grothendieck, "Esquisse d'un Programme,". https://webusers.imj-prg.fr/~leila.schneps/grothendieckcircle/EsquisseFr.pdf.

[83] S. Franco, A. Hanany, D. Martelli, J. Sparks, D. Vegh, and B. Wecht, "Gauge theories from toric geometry and brane tilings," *JHEP* **01** (2006) 128, arXiv:hep-th/0505211.

[84] A. Hanany, Y.-H. He, V. Jejjala, J. Pasukonis, S. Ramgoolam, and D. Rodriguez-Gomez, "The Beta Ansatz: A Tale of Two Complex Structures," *JHEP* **06** (2011) 056, arXiv:1104.5490 [hep-th].

[85] V. Jejjala, S. Ramgoolam, and D. Rodriguez-Gomez, "Toric CFTs, Permutation Triples and Belyi Pairs," *JHEP* **03** (2011) 065, arXiv:1012.2351 [hep-th].

[86] J. Bao, Y.-H. He, and A. Zahabi, "Reflexions on Mahler: Dessins, Modularity and Gauge Theories," arXiv:2111.03655 [hep-th].

[87] Y.-H. He and J. McKay, "N=2 Gauge Theories: Congruence Subgroups, Coset Graphs and Modular Surfaces," *J. Math. Phys.* **54** (2013) 012301, arXiv:1201.3633 [hep-th].

[88] Y.-H. He, J. McKay, and J. Read, "Modular Subgroups, Dessins d'Enfants and Elliptic K3 Surfaces," *LMS J. Comp. Math.* **16** (2013) 271–318, arXiv:1211.1931 [math.AG].

[89] S. K. Ashok, F. Cachazo, and E. Dell'Aquila, "Children's drawings from Seiberg-Witten curves," *Commun. Num. Theor. Phys.* **1** (2007) 237–305, arXiv:hep-th/0611082.

[90] J. Bao, O. Foda, Y.-H. He, E. Hirst, J. Read, Y. Xiao, and F. Yagi, "Dessins d'enfants, Seiberg-Witten curves and conformal blocks," *JHEP* **05** (2021) 065, arXiv:2101.08843 [hep-th].

[91] J. Hultgren, "Permanental Point Processes on Real Tori, Theta Functions and Monge-Ampère Equations," arXiv:1604.05645 [math.AP].

[92] J. Hultgren and M. Önnheim, "An optimal transport approach to Monge-Ampère equations on compact Hessian manifolds," arXiv:1607.02923 [math.DG].

[93] J. Hultgren and D. Witt Nyström, "Coupled Kähler-Einstein metrics," arXiv:1608.07209 [math.DG].

[94] J. Hultgren, "Coupled Kähler-Ricci solitons on toric Fano manifolds," arXiv:1711.09881 [math.DG].

[95] R. J. Berman, "Statistical mechanics of permanents, real-Monge-Ampere equations and optimal transport," arXiv:1302.4045 [math.AP].

[96] S.-Y. Cheng and S.-T. Yau, "On the real monge-ampère equation and affine flat structures," *Proceedings of the 1980 Beijing Syposium on Differential Geometry and Differential Equations* **1-3** (1982) pp339–370.

[97] P. Aspinwall, *Dirichlet Branes and Mirror Symmetry*. Clay mathematics monographs. American Mathematical Society, 2009.

[98] I. Goodfellow *et al.*, "Generative adversarial nets," *Advances in neural information processing systems* **27** (2014) , arXiv:1406.2661 [stat.ML].

[99] N. Lei, K. Su, L. Cui, S. Yau, and X. D. Gu, "A geometric view of optimal transportation and generative model," *CoRR* (2017) , arXiv:1710.05488.

[100] Z. Cao, "Realizing continual learning through modeling a learning system as a fiber bundle," *CoRR* (2019) , arXiv:1903.03511.