Physics Letters B 815 (2021) 136139

Contents lists available at ScienceDirect

Physics Letters B

www.elsevier.com/locate/physletb

Machine learning Calabi-Yau four-folds

Yang-Hui He [a,b,c,*], Andre Lukas [d]

[a] Department of Mathematics, City, University of London, London EC1V 0HB, UK
[b] Merton College, University of Oxford, OX1 4JD, UK
[c] School of Physics, NanKai University, Tianjin, 300071, PR China
[d] Rudolf Peierls Centre for Theoretical Physics, University of Oxford, Parks Road, Oxford OX1 3PU, UK

ARTICLE INFO

Article history:
Received 14 September 2020
Received in revised form 22 December 2020
Accepted 10 February 2021
Available online 12 February 2021
Editor: N. Lambert

ABSTRACT

Hodge numbers of Calabi-Yau manifolds depend non-trivially on the underlying manifold data and they present an interesting challenge for machine learning. In this letter we consider the data set of complete intersection Calabi-Yau four-folds, a set of about 900, 000 topological types, and study supervised learning of the Hodge numbers $h^{1,1}$ and $h^{3,1}$ for these manifolds. We find that $h^{1,1}$ can be successfully learned (to 96% precision) by fully connected classifier and regressor networks. While both types of networks fail for $h^{3,1}$, we show that a more complicated two-branch network, combined with feature enhancement, can act as an efficient regressor (to 98% precision) for $h^{3,1}$, at least for a subset of the data. This hints at the existence of an, as yet unknown, formula for Hodge numbers.

1. Introduction

Topological quantities of manifolds, such as Betti or Hodge numbers, are often non-trivially related to the data describing the underlying manifold and tend to be difficult to work out. Explicit formulae are usually not known and calculations rely on complicated and frequently computationally intense algorithms (see, for example, the volume [1] and references therein for applications of computational algebraic geometry to string and gauge theories). For this reason, such topological properties are an interesting and challenging playground for machine learning. At the most basic level, we can ask if neural networks are capable of learning these properties. In this letter, we will address this problem for complete intersection Calabi-Yau (CICY) four-folds and their Hodge numbers.

The complete set of CICY three-folds was the first large dataset of Calabi-Yau manifolds to be constructed [2,3]. It consists of 7890 different topological types of manifolds which have provided string theorists and mathematicians alike with a fertile ground for exploration (for some recent applications in the context of string theory, see, for example, Refs. [4–8]). More recently, techniques of machine learning have been applied to the study of the string landscape [9–14] (for reviews see Refs. [15,16]). In fact, CICY three-folds were the first data set to be analysed from this viewpoint [9]. Subsequent work has studied Hodge numbers of CICY three-folds systematically, using different types of neural network architectures [17–20].

With the advent of F-theory, Calabi-Yau four-folds have become increasingly important for string compactifications. CICY four-folds have been classified more recently [22] and their relevant topological properties have been computed in Ref. [23]. The dataset is considerably larger and richer than the one for CICY three-folds and it consists of about 900000 topological types of manifolds. However, so far, this new dataset has not been used for machine learning and the purpose of this letter is to fill this gap. More specifically, we will explore, within the context of supervised learning, if and to what extent Hodge numbers of CICY four-folds can be learned by neural networks.

2. Background and notation

2.1. CICY four-folds

A CICY four-fold is defined as a complete intersection of the zero loci of K multi-homogeneous polynomials in the ambient space $A = \mathbb{P}^{n_1} \times \mathbb{P}^{n_2} \times \ldots \times \mathbb{P}^{n_m}$ with dimension $d = n_1 + \cdots + n_m = K + 4$. The degrees of these polynomials are collected in a $m \times K$ *configuration matrix* $Q = (q^i{}_a)$, where $i = 1, \ldots, m$ and $a = 1, \ldots, K$. Its entries $q^i{}_a \in \mathbb{Z}^{\geq 0}$ specify the degree of homogeneity of the a^{th} defining polynomial in the homogeneous coordinates of the i^{th} projective ambient space factor. The Calabi-Yau condition

* Corresponding author.
E-mail addresses: hey@maths.ox.ac.uk (Y.-H. He), andre.lukas@physics.ox.ac.uk (A. Lukas).

https://doi.org/10.1016/j.physletb.2021.136139

$$n_i + 1 = \sum_{a=1}^{K} q^i{}_a \quad \text{for} \quad i = 1, \ldots, m \,, \tag{2.1}$$

fixes the dimensions n_i of the projective spaces in terms of the configuration matrix. A configuration matrix Q, therefore, determines the ambient space A as well as a family of CICY four-folds therein, defined by all (sufficiently generic) polynomials with the specified multi-degrees. Fortunately, many topological quantities, including Hodge numbers, only depend on the family and, hence, only the configuration matrix Q, rather than the specific choice of polynomials.

Calabi-Yau three-folds have two non-trivial[1] Hodge numbers, $h^{1,1}$ and $h^{2,1}$, which are related to the Euler number χ by $\chi = 2(h^{1,1} - h^{2,1})$. Since the Euler number is usually easily determined, three-folds require only one non-trivial Hodge number computation. The situation is significantly more complicated for four-folds which have four non-trivial Hodge numbers, $h^{1,1}$, $h^{2,1}$, $h^{3,1}$ and $h^{2,2}$. In addition to the formula for the Euler number

$$\chi = 6(8 + h^{1,1} + h^{3,1} - h^{2,1}) \,, \tag{2.2}$$

there is an additional linear relation [25]

$$h^{2,2} = 2(22 + 2h^{1,1} + 2h^{3,1} - h^{2,1}) \,, \tag{2.3}$$

between those Hodge numbers which can be derived from the index theorem. As for three-folds, the Euler number is usually easily computed. In fact, for CICYs of any dimension it can be expressed explicitly in terms of the entries of the configuration matrix Q (see, for example, Ref. [26]). In view of Eq. (2.3), this leaves us with two Hodge numbers to be determined by a non-trivial computation and, for our purposes, we will take these to be $h^{1,1}$ and $h^{3,1}$.

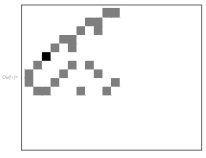
CICY four-folds for which the entire second cohomology descends from the ambient space are called *favourable* and a significant fraction of CICY four-folds have this property. Evidently, favourable four-folds satisfy $h^{1,1} = m$ so in this case one of the remaining Hodge number computations is simple.

2.2. Data sets

The different topological types of CICY four-folds were classified in Ref. [22] (q.v. [23,24]) by listing their configuration matrices. Discarding cases which correspond to direct product manifolds, this has led to 905684 inequivalent configuration matrices Q with minimal size $(m, K) = (1, 1)$ and maximal size $(m, K) = (16, 20)$. About 54% of these are favourable. The distribution of Hodge numbers $h^{1,1}$ and $h^{3,1}$ for this data set is shown in Fig. 1.

The configuration matrices have different sizes so, as stands, they are not well-suited for training neural networks. We resolve this problem by padding each configuration Q with zeros (on the right and at the bottom) to create a 16×20 enhanced configuration matrix $\tilde{Q}$, whose size matches that of the largest configuration. As an example, the enhanced configuration matrix $\tilde{Q}$ for a configuration Q with $(m, K) = (10, 12)$ is given by

$$\tilde{Q} = \begin{pmatrix}
0000000001100000000 \\
0000000110000000000 \\
0000001010000000000 \\
0000110000000000000 \\
0001010000000000000 \\
0020000000000000000 \\
0100010100000000000 \\
1000100010000000000 \\
1001000001000000000 \\
0110001001000000000 \\
0000000000000000000 \\
0000000000000000000 \\
0000000000000000000 \\
0000000000000000000 \\
0000000000000000000 \\
0000000000000000000
\end{pmatrix} \leftrightarrow$$



[1] Throughout this letter, we only consider smooth, compact Calabi-Yau n-folds with holonomy group $SU(n)$.

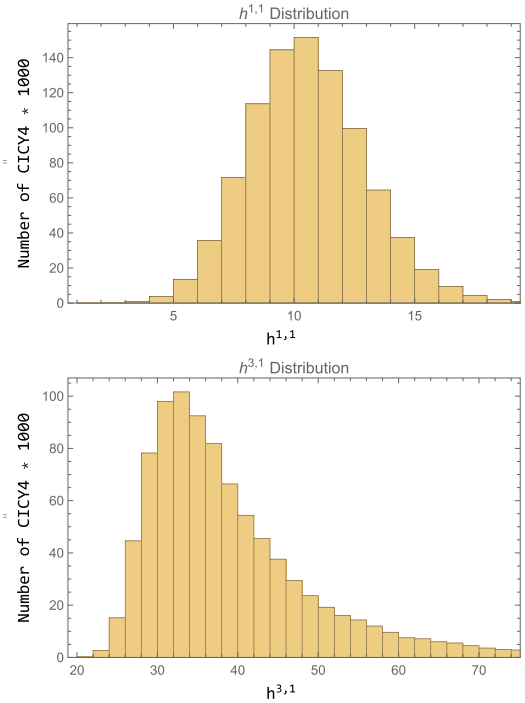


Fig. 1. Distribution of $h^{1,1}$ and $h^{3,1}$ for all CICY four-folds.

On the right, we have represented $\tilde{Q}$ by an image with 16×20 pixels and the typical entries $0, 1, 2$ of $\tilde{Q}$ mapped to grey-scales. This has been done, as was in Ref. [9], to emphasise the analogy of our problem with pattern recognition. However, unlike for typical pattern recognition problems (such as classifying the MNIST numbers), it is not intuitively clear how our target (the Hodge numbers) is related to the features of the image.

Our data sets will be of the form

$$D^{1,1} = \{\tilde{Q} \to h^{1,1}\} \quad \text{or} \quad D^{3,1} = \{\tilde{Q} \to h^{3,1}\} \,. \tag{2.4}$$

It is possible to enlarge these data sets by adding equivalent configurations obtained from the given ones by simultaneous permutations of rows and columns. Indeed, this method has been used to enlarge the set of CICY three-folds in Refs. [9,10,17,18]. However, the set of CICY four-folds is considerably larger and numbers are certainly sufficient for machine learning purposes without any enlargement. In fact, for cases where we use the entire data set we have also checked that enlarging by equivalent configurations does not significantly increase the performance of the networks we study. For these reasons, we take the data sets $D^{1,1}$ and $D^{3,1}$ above to contain the 905684 inequivalent (enhanced) configurations of the original classification.

We will also analyse *feature-enhanced* versions of these data sets where we supplement the configuration matrix Q by monomials of degree up to four in its entries $q^i{}_a$. For larger configuration matrices this leads to very large input spaces which are not practical. For this reason and for specificity we will limit our discussion to configurations with size $(m, K) = (4, 4)$. This subset only contains 1035 configurations so, unlike for the full data set, we now opt for an enlargement by all simultaneous row and column permutations of Q. This leads to a total of $1035 \times 24^2 = 596160$ configurations. These 4×4 configurations Q are then feature-enhanced to a vector Q_q, where $q = 1, 2, 3, 4$, by including all monomials of degree $\leq q$ between the column entries of Q. For $q = 2$ this means $Q_2 = (q^i{}_a, q^j{}_a q^k{}_a)$, where $a, i, j, k = 1, \ldots, 4$ and

$j \leq k$, and analogously for $q > 2$. The dimensions[2] d_q of these enhanced configurations are given by $(d_1, d_2, d_3, d_4) = (4 \times 4, 14 \times 4, 34 \times 4, 69 \times 4)$. This choice of feature-enhancement is inspired by the success in bundle-cohomology calculations [14,30,31] and well-motivated mathematically. For an n-fold, cohomology, and Hodge numbers in particular, should depend on n-tuple intersection of divisors, and thus on polynomials up to degree n in the multi-degrees of the bundles; hence we add this information to the input.[3] In summary, this leads to data sets of the form

$$D_q^{1,1} = \{Q_q \to h^{1,1}\} \quad \text{or} \quad D_q^{3,1} = \{Q_q \to h^{3,1}\}\,, \tag{2.5}$$

where $q = 1, 2, 3, 4$. As is customary, we need to *disjointly* split the above data sets into a training set, a validation set and a test set. We typically use 15% for training and 5% for validation, both randomly selected from the full set, and the remainder of 80% for testing. The validation set is used to monitor progress during training and we evaluate the trained network on the test set.

2.3. Neural networks

Key components in the subsequent discussion are standard forward-feed, fully connected neural networks of depth d, which define a map of the form

$$\mathbb{R}^{n_0} \xrightarrow{L_{n_1}} \mathbb{R}^{n_1} \xrightarrow{f} \mathbb{R}^{n_1} \xrightarrow{L_{n_2}} \cdots \xrightarrow{L_{n_d}} \mathbb{R}^{n_d} \xrightarrow{f} \mathbb{R}^{n_d}\,.$$

Here L_n is a standard affine transformation with trainable weights and biases and co-domain dimension n and f represents a component-wise function, typically a logistic sigmoid function, $\sigma(z) := (1 + e^{-z})^{-1}$, or a scaled exponential-linear unit (SELU), defined by $s(z) = 1.0507\,z$ for $z \geq 0$ and $s(z) = 1.7851(\exp(z) - 1)$ for $z < 0$. In some cases, we will use a probability p dropout layer, denoted δ_p, which is a standard tool to avoid over-fitting. The dropout probability p is chosen to optimise performance. For classifier networks we also require a softmax layer $S(z_i) = e^{z_i}(\sum_i e^{z_i})^{-1}$. For notational convenience, we will use the shorthand $\mathcal{N}_{n_0}(n_1, f, n_2, \ldots, n_d, f)$ for the above network.

Explicit training is carried out with the Mathematica machine learning suite [32], using the ADAM [33] steepest gradient descent minimiser and a mean square loss (for the categorical classification cases we also tried cross-entropy loss and the results were comparable). Evidently, the network architectures explored in this letter are relatively simple. We have checked that convolutional networks, similar to those used for digit recognition, do not improve the performance significantly. However, it would be expedient to apply the methods of Ref. [19,20], as well as the interesting representation of configurations in [21] to the CICY four-fold data set.

3. Learning $h^{1,1}$

Fig. 1 shows that $h^{1,1}$ takes a rather limited set of values for our data set. More specifically, it turns out that $h^{1,1} \in \{1, 2, \ldots, 24\}$. This suggests that both a 24-way classifier network and a regressor network with a real output intended as an approximation of $h^{1,1}$ may be feasible. We discuss these two options in turn.

[2] Recall that one can write $\binom{m+d-1}{d}$ independent monomials of degree d in m variables (here $m = 4$), so there are 10 quadrics, 20 cubics, and 35 quartics composed from the columns of Q.

[3] There are, indeed, more equivalences of configurations than just permutations and are more sophisticated mathematical equivalences such as splitting. In the threefold case such enhancement were performed in [21]. However, for the four-folds, such data is not readily available so we do not consider them in this work.

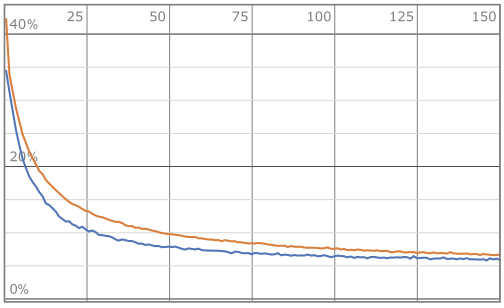


Fig. 2. The training plot for the data set $D^{1,1}$ in Eq. (2.4) and the classifier network (3.1). Indicated is the error rate as a function of training rounds for the training set (orange) and the validation set (blue).

3.1. Classifier network

The relevant data set for this task is $D^{1,1}$ in Eq. (2.4) which is used to train a network of the form

$$N_{16\times 20}(512, \sigma, \delta_{0.4}, 256, \sigma, \delta_{0.3}, 256, \sigma, 24, S)\,. \tag{3.1}$$

As mentioned earlier, we use 15% of the data set for training and 5% for validation. Training is performed at a learning rate of 1/300 for 150 rounds and takes about 18 minutes on a single laptop CPU. The training curves are shown in Fig. 2. The trained network is applied to the test set (at 80% the bulk of the data) and it predicts $h^{1,1}$ correctly for 96% of the cases. The 24×24 confusion matrix is diagonal to a good accuracy, with any single off-diagonal entry < 0.05. This is a rather convincing performance by a relatively simple, feed-forward network. We note that the substantial width of the network (3.1) is required to achieve the stated accuracy and we have to include the dropout layers in order to avoid over-fitting. In summary, we conclude that the Hodge numbers $h^{1,1}$ for CICY four-folds can be successfully learned by a suitably configured fully connected classifier network.

Not surprisingly, for favourable manifolds, the network predicts $h^{1,1}$ with 100% accuracy, so misclassifications only arise for non-favourable cases. This observation suggests that a simple binary classifier network, similar to (3.1) but with the 24-dimensional output layer replaced by a two-dimensional one, can be used to distinguish favourable and non-favourable CICY four-folds. This is indeed the case and works at about 96% accuracy on the test set.

The above network generalises well when trained on a randomly selected training set. A somewhat more ambitious question is whether a network trained on configurations with small Hodge number, say $h^{1,1} < 8$ (about 20% of the configurations), can predict the Hodge numbers of configurations with $h^{1,1} \geq 8$. For CICY three-folds this was attempted in Ref. [18]. Obviously, such a network, trained only on small and relatively simple configurations but able to predict properties of larger and more complicated ones would be very useful. Unfortunately, for the case of CICY four-folds and classifier networks of the type (3.1) this does not work well and the network performs poorly, with a success rate close to zero on configurations with $h^{1,1} \geq 8$. However, seeding the training set with a small sample (say 10000) of configurations with $h^{1,1} \geq 8$ leads to a significant improvement (success rate around 0.6).

3.2. Regressor network

Encouraged by the success of the classifier, let us see how a regressor performs. We emphasize that the difference with the regressor and the classifier is that the former puts the input data into some category while the latter tries to find a "best-fit" function (albeit complicated) that analytically gives the Hodge numbers from the input configuration. This might be more mathematically

interesting than a mere categorization. We use the same dataset $D^{1,1}$ in Eq. (2.4) and a network of the form

$$\mathcal{N}_{16\times 20}(512, s, 256, s, 128, s, 32, s, 8, s, 1)\ . \tag{3.2}$$

The idea is that the one-dimensional real output of this network approximates $h^{1,1}$ and we take its rounding to the nearest integer as the prediction for $h^{1,1}$. This is clearly challenging since a successful prediction requires an accurately trained network with a typical loss significantly less than one.

The above network is the best-performing we have found. After training for 150 rounds at a learning rate of 1/1000 (about 15 minutes on a single CPU), the network output has an average deviation from $h^{1,1}$ of ~ 0.3 on the test set. This translates, after rounding, into 83% of test set values correctly predicted. While this is a respectable success rate and the network trains efficiently, a wrong prediction for $h^{1,1}$, typically by 1, in 17% of the cases means the network is of limited practical use.

4. Learning $h^{3,1}$

Fig. 1 shows that the range of $h^{3,1}$ values is considerably larger than the one for $h^{1,1}$. More specifically, we have $20 \leq h^{3,1} \leq 426$. As we will see, for this range it is significantly harder to obtain convincing performances from simple classifier or regressor networks of the kind we have used for $h^{1,1}$. For this reason, we also explore other options, focusing on the feature-enhanced data sets $D_q^{3,1}$ in Eq. (2.5) and more complicated two-branch networks.

4.1. Classifier and regressor networks

A 407-way classifier based on a network of the form

$$N_{16\times 20}(512, \sigma, \delta_{0.4}, 512, \sigma, \delta_{0.4}, 512, \sigma, \delta_{0.4}, 407, S)$$

trained on the dataset $D^{3,1}$ in Eq. (2.4) leads to a poor performance, with a 27% success rate on the test set. Likewise, a regressor network of the form

$$N_{16\times 20}(256, s, \delta_{0.2}, 128, s, 16, s, 1)\ ,$$

trained on $D^{3,1}$, produces test set predications for $h^{3,1}$ with an average deviation of ~ 2.7 from the true value. While this might be considered a reasonable accuracy for some purposes, it is not sufficient to predict the correct integer after rounding. In fact, only 15% of test set values for $h^{3,1}$ are reproduced exactly after rounding. For either of the above networks, we have not been able to improve performance significantly by hyper-parameter optimisation.

4.2. Classifier and regressor for 4×4 configurations

We can ask if a classifier network performs better on the data set $D_1^{3,1}$ of 4×4 configurations as defined in Eq. (2.5). In addition to a much smaller dimension of the feature space, the range of $h^{3,1}$ values is now reduced to $20 \leq h^{3,1} \leq 260$. In fact, a 235-way classifier network of the form

$$\mathcal{N}_{4\times 4}(512, \sigma, \delta_{0.4}, 512, \sigma, \delta_{0.4}, 512, \sigma, \delta_{0.4}, 235, S)\ ,$$

trained on $D_1^{3,1}$ performs perfectly on the test set at a 100% success rate. This is quite impressive, considering the number of classes is still large.

However, a regressor network of the form

$$\mathcal{N}_{4\times 4}(512, s, 256, s, 64, s, 16, s, 1)$$

trained on $D_1^{3,1}$ is much less successful. It predicts $h^{3,1}$ for the test set with an average error of ~ 1 which leads to a success rate of 35% after rounding. We have not been able to improve this performance significantly by variations in hyper-parameters.

4.3. Two branch network and feature enhancement

Is it possible to construct a successful regressor network for $h^{3,1}$? The approach we are about to present is motivated by observations made in the related context of line bundle cohomology. Line bundle cohomology dimensions have been conjectured, in many cases empirically verified [27–29] and for some classes shown [30] to be described by piecewise polynomial formulae in the line bundle degrees. The degree of the polynomials equals the complex dimension of the underlying manifold. In Ref. [31] a two-branch neural network adapted to this structure and trained with feature-enhanced data has been constructed. It has been shown that conjectures for piecewise polynomial cohomology formulae can be extracted from this network.

The present context is of course somewhat different. We are not interested in all line bundles on a fixed manifold but rather in specific properties for a class of different manifolds. Nevertheless, it is the case that computations of Hodge numbers for CICYs are ultimately reduced to the computation of line bundle cohomology. For this reason it is not far-fetched to try a two-branch regressor network, similar to the one used in Ref. [31], in order to learn $h^{3,1}$.

More specifically, we would like to consider networks of the form

$$\begin{array}{rcl} Q_1 & \to & \mathcal{N}_{4\times 4}(512, \sigma, 512, \sigma, 256, \sigma) \searrow \\ & & \qquad\qquad\qquad\qquad \text{dot} \to h^{3,1} \\ Q_q & \to & \mathcal{N}_{d_q}(256) \nearrow \end{array}$$

where "dot" indicates a dot product between the two vectors. The upper branch of the network is intended to detect the regions of the underlying piecewise polynomial formula. Since the boundaries of these regions are usually described by linear equations the upper branch only receives the 4×4 configuration matrices $Q_1 \in D_1^{3,1}$. On the other hand, the lower part of the network, which consists of a single affine layer is supposed to reproduce the polynomial and, therefore, receives the feature-enhanced matrices $Q_q \in D_q^{3,1}$ which consists of the configuration matrix as well as its monomials of degree $\leq q$. The analogy with line bundles suggests that we need up to quartic monomials (since we are working on four-folds). We have, therefore, constructed the data sets $D_q^{3,1}$ for q up to four. For comparison purposes we will consider all cases $q = 1, 2, 3, 4$.

Training the above network with $D_1^{3,1}$ and $D_q^{3,1}$ for $q = 1, 2$ leads to poor performance, with an average error of ~ 1 and a test set success rate of 54% for $q = 1$ and 40% for $q = 2$. On the other hand, training with $D_1^{3,1}$ and $D_q^{3,1}$ for $q = 3, 4$ leads to very accurately trained networks. Specifically, for $q = 3$ we achieve an average error of 0.04 which translates into a 98% success rate on the test set. For $q = 4$ the results are similar, with an average error of 0.17 and a test set success rate of 95%.

Achieving this accuracy for $q = 3, 4$ requires a careful adjustment of the learning rate during training. In an initial training step of about 100 rounds the learning rate is set to 1/1000. This leads to a network whose average error does not decrease below ~ 1. Adding successive short training steps of about 5 - 10 rounds with gradually decreasing learning rate to a final value of 1/100000 then leads to the accuracy mentioned above.

In conclusion, we are able to build a successful regressor for $h^{3,1}$, at least for the 4×4 configurations under consideration, by using a two branch network motivated by the results for line bundle cohomology in Ref. [31]. As expected, we require feature-enhanced data which includes at least quadrics and cubics of the configuration matrix for this network to perform well. We note that the two-branch network can also be applied to the data sets $D_q^{1,1}$ for $h^{1,1}$, where it leads to a 100% success rate on the test set.

5. Conclusion & outlook

Computing Hodge numbers of Calabi-Yau manifolds is a non-trivial task and presently known methods require, in all but special cases, complicated algorithms in commutative algebra, based on sequence-chasing in cohomology. For this reason, machine learning of Hodge numbers is an interesting and challenging task. This problem has obvious analogies with image classification, as originally pointed out in Ref. [9]. Despite this analogy, it is, a priori, unclear if these numbers can be successfully learned and, if so, what the required network architectures might be. Indeed, from universal approximation theorem, we expect that an NN could fit the given data of Hodge numbers. However, it is not clear that we could extrapolate from the training data to truly learn some underlying functional dependence.

In this letter, we have studied supervised machine learning of the Hodge numbers $h^{1,1}$ and $h^{3,1}$ for complete intersection Calabi-Yau (CICY) four-folds. This data set consists of about 900,000 topological types, each described by an integer (configuration) matrix Q. We find that $h^{1,1}$ can be successfully predicted from Q with both fully connected classifier and regressor networks. The former are particularly effective and lead to a 96% success rate on the test set when trained on only 15% of the data.

Unfortunately, fully connected classifier or regressor networks do not work efficiently for $h^{3,1}$, presumably due to the large range of $h^{3,1}$ values. However, we have shown that a two branch regressor network, combined with feature enhanced data, works well for a subset of the data which consists of 4×4 configuration matrices.

The structure of this two-branch network is motivated by recent results for line bundle cohomology [27–30]. Its success hints at the existence of a formula for $h^{3,1}$ in terms of Q which is at present unknown. It would be interesting to search for this formula, possibly assisted by the information encoded in the trained network. Such a formula would be a new mathematical result and useful for applications in theoretical physics.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

YHH would like to thank STFC UK, for grant ST/J00037X/1. AL would like to thank Andrei Constantin for discussions.

References

[1] Y-H. He, P. Candelas, A. Hanany, A. Lukas, B. Ovrut, Computational algebraic geometry in string and gauge theory, www.hindawi.com/journals/ahep/2012/431898/.
[2] P. Candelas, A.M. Dale, C.A. Lutken, R. Schimmrigk, Complete intersection CY, Nucl. Phys. B 298 (1988) 493, https://doi.org/10.1016/0550-3213(88)90352-5.
[3] P.S. Green, T. Hubsch, C.A. Lutken, All Hodge numbers of all complete intersection Calabi-Yau manifolds, Class. Quantum Gravity 6 (1989) 105–124, https://doi.org/10.1088/0264-9381/6/2/006.
[4] V. Braun, On free quotients of complete intersection CY, J. High Energy Phys. 04 (2011) 005, arXiv:1003.3235.
[5] A. Lukas, C. Mishra, Discrete symmetries of complete intersection Calabi-Yau manifolds, arXiv:1708.08943.
[6] L.B. Anderson, X. Gao, J. Gray, S.J. Lee, Fibrations in CICY 3-folds, J. High Energy Phys. 10 (2017) 077, arXiv:1708.07907.
[7] L.B. Anderson, Y.H. He, A. Lukas, Heterotic compactification, an algorithmic approach, J. High Energy Phys. 07 (2007) 049, arXiv:hep-th/0702210.
[8] L.B. Anderson, A. Constantin, J. Gray, A. Lukas, E. Palti, A comprehensive scan for heterotic SU(5) GUT models, J. High Energy Phys. 01 (2014) 047, arXiv:1307.4787.
[9] Y.H. He, Deep-learning the landscape, arXiv:1706.02714.
[10] Y.H. He, Machine-learning the string landscape, Phys. Lett. B 774 (2017) 564.
[11] D. Krefl, R.K. Seong, Machine learning of Calabi-Yau volumes, Phys. Rev. D 96 (6) (2017) 066014, arXiv:1706.03346.
[12] F. Ruehle, Evolving neural networks with genetic algorithms to study the string landscape, J. High Energy Phys. 1708 (2017) 038, arXiv:1706.07024.
[13] J. Carifio, J. Halverson, D. Krioukov, B.D. Nelson, Machine learning in the string landscape, J. High Energy Phys. 1709 (2017) 157, arXiv:1707.00655.
[14] R. Deen, Y.H. He, S.J. Lee, A. Lukas, Machine learning string standard models, arXiv:2003.13339.
[15] Y.H. He, The Calabi-Yau landscape: from geometry, to physics, to machine-learning, arXiv:1812.02893.
[16] F. Ruehle, Data science applications to string theory, Phys. Rep. 839 (2020) 1.
[17] K. Bull, Y.H. He, V. Jejjala, C. Mishra, Machine learning CICY threefolds, Phys. Lett. B 785 (2018) 65–72, arXiv:1806.03121.
[18] K. Bull, Y.H. He, V. Jejjala, C. Mishra, Getting CICY high, Phys. Lett. B 795 (2019) 700–706, arXiv:1903.03113.
[19] H. Erbin, R. Finotello, Inception neural network for complete intersection Calabi-Yau 3-folds, arXiv:2007.13379.
[20] H. Erbin, R. Finotello, ML for CICYs: a methodological study, arXiv:2007.15706.
[21] S. Krippendorf, M. Syvaeri, Detecting symmetries with neural networks, arXiv:2003.13679.
[22] J. Gray, A.S. Haupt, A. Lukas, All CICY four-folds, J. High Energy Phys. 07 (2013) 070, arXiv:1303.1832.
[23] J. Gray, A.S. Haupt, A. Lukas, Topological invariants and fibration structure of complete intersection Calabi-Yau four-folds, J. High Energy Phys. 09 (2014) 093, arXiv:1405.2073.
[24] Y. Kimura, Discrete gauge groups in F-theory models on genus-one fibered Calabi-Yau 4-folds without section, J. High Energy Phys. 04 (2017) 168, arXiv:1608.07219.
[25] S. Sethi, C. Vafa, E. Witten, Constraints on low dimensional string compactifications, Nucl. Phys. B 480 (1996) 213–224, arXiv:hep-th/9606122.
[26] T. Hübsch, CY manifolds: a bestiary for physicists, www.worldscientific.com/worldscibooks/10.1142/1410.
[27] A. Constantin, A. Lukas, Formulae for line bundle cohomology on Calabi–Yau threefolds, Fortschr. Phys. 67 (12) (2019) 1900084, arXiv:1808.09992.
[28] D. Klaewer, L. Schlechter, Machine learning line bundle cohomologies of hypersurfaces in toric varieties, Phys. Lett. B 789 (2019) 438–443, arXiv:1809.02547.
[29] M. Larfors, R. Schneider, Line bundle cohomologies on CICYs with Picard number two, Fortschr. Phys. 67 (12) (2019) 1900083, arXiv:1906.00392.
[30] C.R. Brodie, A. Constantin, R. Deen, A. Lukas, Index formulae for line bundle cohomology on complex surfaces, Fortschr. Phys. 68 (2) (2020) 1900086, arXiv:1906.08769.
[31] C.R. Brodie, A. Constantin, R. Deen, A. Lukas, Machine learning line bundle cohomology, Fortschr. Phys. 68 (1) (2020) 1900087, arXiv:1906.08730.
[32] Wolfram Research, Inc., Mathematica, 12.1.0, Champaign, IL, https://www.wolfram.com/mathematica/, 2020.
[33] Diederik P. Kingma, Jimmy Ba, Adam: a method for stochastic optimization, arXiv:1412.6980.