Physics Letters B 832 (2022) 137213

Contents lists available at ScienceDirect

Physics Letters B

www.elsevier.com/locate/physletb

Machine-learning the classification of spacetimes

Yang-Hui He [a,b,c,d], Juan Manuel Pérez Ipiña [e,*]

[a] London Institute, Royal Institution of GB, 21 Albemarle St., London, W1S 4BS, UK
[b] Merton College, University of Oxford, OX1 4JD, UK
[c] Department of Mathematics, City, University of London, London EC1V0HB, UK
[d] School of Physics, NanKai University, Tianjin, 300071, PR China
[e] Mathematical Institute, University of Oxford, Andrew Wiles Building, Radcliffe Observatory Quarter, Woodstock Road, Oxford, OX2 6GG, UK

ARTICLE INFO

Article history:
Received 18 January 2022
Received in revised form 7 April 2022
Accepted 26 May 2022
Available online 31 May 2022
Editor: A. Volovich

ABSTRACT

On the long-established classification problems in general relativity we take a novel perspective by adopting fruitful techniques from machine learning and modern data-science. In particular, we model Petrov's classification of spacetimes, and show that a feed-forward neural network can achieve high degree of success. We also show how data visualization techniques with dimensionality reduction can help analyze the underlying patterns in the structure of the different types of spacetimes.

1. Introduction & summary

What are the possible structures of spacetime? This is surely one of the most important questions in theoretical physics. Classification problems in general relativity have been an active field since the very beginning and have more recently been a focus of computer algebra systems [1,2]. Fully classifying and comparing Riemannian manifolds can be achieved through the Cartan-Karlhede algorithm [3]. The first step in this algorithm is to determine the Petrov [4] and Segre [5] types of the spacetime [1,6]. These methods analyze algebraic symmetries of the Weyl and Ricci tensor, respectively, and involve detailed study of roots and multiplicities of certain quartic equations. In particular, Petrov's classification of the Weyl tensor has been an integral part of the study of exact solutions to the Einstein equations. Here, we will illustrate a new computational approach that can be used in the Petrov classification problem, which can then be extended for a full classification of gravitational solutions.

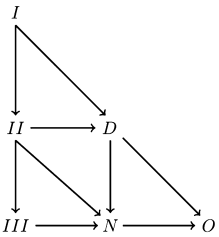
Since the recent introduction of machine-learning and related techniques of modern data science, to study the string theory landscape [7–11], and more generally the vast landscape of pure mathematics [12–16], it is natural to address our present problem of spacetime classification under the auspices of this programme. The reader is also referred to the pedagogical introduction of machine-learning in theoretical physics and mathematics by [17,18] as well as references therein. Furthermore, detection of symmetries in physical systems relevant to our context, using machine-learning, is also discussed in [19–25].

In this letter, we apply some of these machine-learning (ML) techniques to Petrov's classification of spacetimes. Since the original formulation of the problem, many algorithms have been proposed to model the classification (see, for example, [6,26–29]). These usually reduce the problem to finding the roots and multiplicites of a quartic equation where the parameters are a set of five complex Weyl scalars Ψ_i $(i = 0, \ldots, 4)$ in the Newman-Penrose formalism [30]. These Weyl scalars can easily be computed for any spacetime and the relations between the nonvanishing Ψ_i determine the Petrov type of the manifold. Here we take this approach for building a supervised learning problem fit for ML tools.

In Section 2 we give an overview of the problem and show how to represent the spacetime data in an expedient manner. We artificially generate numerical data to train and validate various ML classifiers. Specifically, we start by building different datasets of Weyl scalars $\{\Psi_0, \Psi_1, \Psi_2, \Psi_3, \Psi_4\}$, with randomly generated entries, and then manually labelling each data point with its corresponding Petrov type. These datasets are later used in Section 3 to train several ML classifiers to see how well they learn and compare. We find that feed-forward neural networks (NN) are the most accurate classifiers for this problem, obtaining very high precision in only a handful of epochs. Moreover, in Section 4, we use other data science techniques, like Principal Component Analysis (PCA), to further study latent patterns in the data, that give rise to the Petrov classification. We show how data visualization tools can illustrate the intrinsic differences between spacetimes of distinct Petrov type. Finally, we discuss the results and future applications of this programme in Section 5.

* Corresponding author.
E-mail addresses: hey@maths.ox.ac.uk (Y.-H. He), Juan.PerezIpina@maths.ox.ac.uk (J.M. Pérez Ipiña).

https://doi.org/10.1016/j.physletb.2022.137213



Petrov type	Multiplicities
I	(1,1,1,1)
II	(2,1,1)
D	(2,2)
III	(3,1)
N	(4)
O	-

Fig. 1. Classification of the Petrov type according to the number and multiplicity of principal null directions (with the arrows denoting possible degenerations of one Petrov type into another). Type O corresponds to the vanishing of the Weyl tensor and so does not single out any principal null directions.

2. The Petrov classification

Petrov's classification of the algebraic symmetries of the Weyl tensor can be formulated as an eigenvalue problem for the Weyl tensor evaluated at some spacetime event. Alternatively, one can see it as a characterization of the Weyl tensor in terms of the principal null directions (p.n.d.) at that event [31] (see the Appendix for details into this approach). Depending on the amount and multiplicity of the p.n.d.'s we can classify spacetimes in 6 distinct types: I, II, III, D, N, O. The classification can be seen in Fig. 1.

As is shown in the Appendix, one can see that the determination of principal null directions is equivalent to solving the following quartic equation for z:

$$\Psi_0 - 4z\Psi_1 + 6z^2\Psi_2 - 4z^3\Psi_3 + z^4\Psi_4 = 0\,, \tag{2.1}$$

where Ψ_i $(i = 0, 1, 2, 3, 4)$ are the five complex Weyl scalars in the Newman-Penrose formalism, and are defined in (A.9).

a. The $n = 32$ cases: The vanishing of one or more of these Ψ_i simplifies (2.1), and this has been a staple of most attempts to determine the Petrov type. This has been taken into account in many of the aforementioned algorithms where, starting with the work of [27], a parameter n was introduced to distinguish the 32 possible combinations of vanishing/non-vanishing Weyl scalars. Each of these 32 classes might have one or more Petrov types assigned to it. For a detailed list of the classes and their Petrov types, see Table 1 (where we've ordered the cases according to the number of vanishing Weyl scalars, and not on the value of n from [27]).

As can be seen from Table 1, for the cases with 3 or more vanishing Weyl scalars the Petrov type can be immediately determined; this is not the case for the rest. When working with an arbitrary null tetrad, the Weyl vector $\{\Psi_i\}$ might be arbitrarily hard and it takes more work to determine the Petrov type. Of course, the Petrov classification is coordinate independent and specifically, does not depend on the choice of tetrad, as long as that frame is not singular [32]. To distinguish between the possible types at the bottom half of the table, there have been many analytical results as in [6,27] (building polynomials out of the remaining non-vanishing Weyl scalars). Since we want our classifier to handle completely general data (and work in any basis), we want to train in all possible cases of Table 1.

b. Data generation: For this purpose we treat $\{\Psi_0, \Psi_1, \Psi_2, \Psi_3, \Psi_4\}$ as a numerical five-vector, randomly generating the entries for every possible case and subcase in Table 1.

We created different databases formed from integer, rational, real or complex entries. The latter two permit the creation of huge datasets uniformly distributed in a specific range (e.g. for the reals $\Psi_i \in \{-10, 10\}$). Unfortunately, for the real and complex data points, some subcases where not possible to sample through purely random generation so the analytical results of [6,27] were used to generate this remaining data.

Table 1
Determination of the Petrov type according to the vanishing of the five Weyl scalars $\{\Psi_0, \Psi_1, \Psi_2, \Psi_3, \Psi_4\}$. "Form" refers to the vanishing of the five quantities Ψ_i: N signifies a non-vanishing entry and 0, a vanishing one. This table was based on the one at [6], where we also corrected some typos.

Number of zeros	Form	Petrov type
5	00000	O
4	N0000	N
	0000N	
	00N00	D
	0N000	III
	000N0	
3	NN000	III
	000NN	
	00N0N	II
	N0N00	
	00NN0	
	0NN00	
	0N00N	I
	N00N0	
	0N0N0	
	N000N	
2	00NNN	II or D
	NNN00	
	N0N0N	I or D
	0N0NN	I or II
	NN0N0	
	N00NN	
	NN00N	
	0NN0N	
	N0NN0	
	0NNN0	
1	0NNNN	I, II or III
	NNNN0	
	N0NNN	
	NNN0N	
	NN0NN	I, II, III or D
0	NNNNN	I, II, III, D or N

Table 2
Tally of data points per Petrov type for the dataset of real entries. The distribution of points per class is not homogeneous and this has to be taken into account when judging the efficiency of a classifier.

I	II	III	D	N	O	Total
126,000	90,500	55,500	24,500	23,500	10,000	330,000

Specifically, for the real (or complex) dataset, 10,000 points were collected from each n except the last case, NNNNN, where 20,000 points were sampled. This amounts to a total size of 330,000 data points, of different Petrov types. Notice that by doing this we are taking a different number of data points per Petrov type but this is consistent with how common it is to find each type. For example, for the real dataset the resulting tally of points per type can be seen in Table 2. These vectors were then labelled by their corresponding Petrov type, through the implementation of the [26] algorithm in Mathematica [33].

3. Building a classifier

For our supervized ML paradigm, the above dataset was randomly split in three groups: 70% for the training set, 15% for validation and 15% for testing. The first two are used to train the classifiers, while the testing set is used to evaluate the perfor-

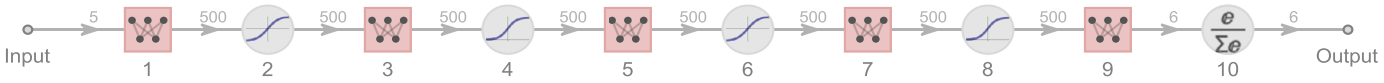


Fig. 2. Architecture of the five-layer neural network. The hidden layers are alternating between tanh and sigmoid activated; the last layer corresponds to the softmax activation function. The hidden layers contain 500 nodes each, and the softmax has 6, corresponding to the six output classes.

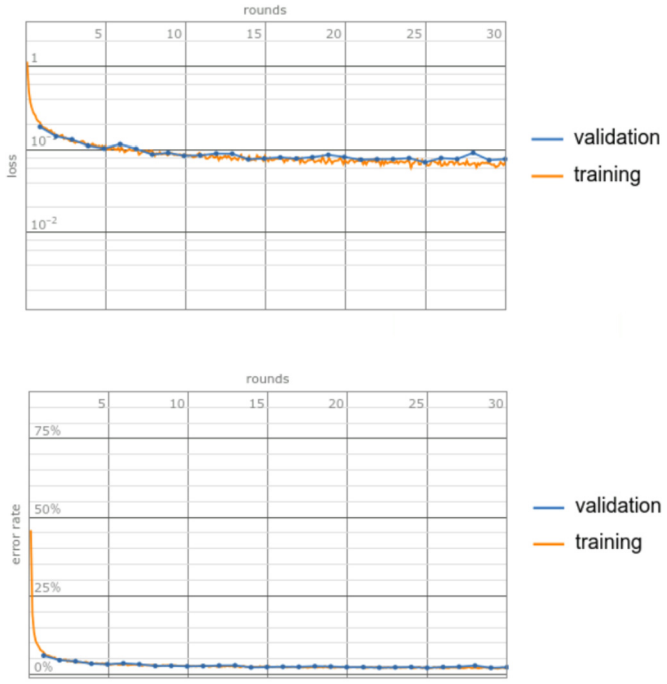


Fig. 3. Loss-function (above) and error rate (below) for the training of the neural network, plotted against the number of epochs or rounds.

mance in never before seen data. Many different types of classifiers were trained and tested, including: decision trees (boosted), random forests, nearest neighbours, and more. While these methods achieved reasonable accuracies, the best results were obtained using feed-forward neural networks (NN), which we detail shortly.

The non-linearity in a NN model is obtained through the choice of activation functions. For this problem we found the highest accuracy in the use of hyperbolic tangents and logistic sigmoids. While the problem can be modelled using a single hidden layer, we found higher accuracy in fewer epochs when using multiple hidden layers. In Fig. 2 we show the architecture of our NN that combines both activation functions in multiple alternating hidden layers. It takes as input the five-dimensional[1] vectors of Weyl numbers, then goes through four hidden layers of 500 nodes each, with alternating activation functions: tanh and sigmoids. The specific numbers of nodes and hidden layers were also found to produce the highest accuracy results, but by no means do we claim this to be the most efficient configuration possible. Different choices of these hyperparameters (or other variables such as the optimizer, the label encoding or the learning rate) represent possible directions of improvements on this neural network. Finally, since we have here a multi-class classification problem, the last layer is a softmax layer, with 6 nodes for the 6 different classes.

As mentioned above, the NN from Fig. 2 was trained and optimized using the training and validation sets, and the testing set was used to determine its accuracy. The network was trained for 30 epochs, using a learning rate of 10^{-3}, and the ADAM optimizer [34]. In Fig. 3 one can see the steady decrease of the loss function and error rate, as the number of training rounds increases. We define accuracy as percentage agreement of predicted versus actual values. However, when dealing with imbalanced multi-class classification problems accuracy is not the most useful evaluation metric. To take these differences into account we define confidence then through the use of Matthew's Correlation Coefficient (MCC) φ generalized to the multi-class case.[2] In all, we achieved an accuracy of 0.979, confidence of 0.973, and a final loss of 6.61×10^{-2}.

We can plot the confusion matrix to see the successes and mistakes for each class. This is a 6×6 integer matrix of the actual numbers in the Petrov class of (O, I, II, III, D, N) versus the numbers as predicted by the ML classifier (Fig. 5).

4. Data visualization

Having successfully trained a neural network (as well as other classifiers) in learning the Weyl data, it is also interesting to see how our data looks, and what patterns we directly observe. This is very much in the spirit of conjecturing formulation via ML [15], to let ML algorithms detect patterns which might ab initio be hidden. For this we can follow a standard Principal Component Analysis (PCA) to dimensionally reduce the data to its highest-variance components so we can study the resulting two-dimensional plots.

We first obtain the principal components from the full unlabelled dataset and then reattach the corresponding labels (colour-coding each distinct class for visualization). In Fig. 4 we can see the principal component representation for each Petrov type (not type O since it corresponds only to the vector $\{0, 0, 0, 0, 0\}$ and therefore has no variation). Note that within the populated areas of the plots, some are more densely populated than others, reflecting the specific data generation procedure of Section 2.

One can see how in the most general case, the data is spread out everywhere with no pattern in sight. As we increase the degeneration (that is, we move downward in Fig. 1, the data starts settling into definite patterns.

In particular, types D and N have very specific shapes, illustrating the particularity of these cases. One can for instance superpose these figures and see exactly how one Petrov type degenerates into another, but for clarity we do not do this since overlaying will obscure many points in the plot.

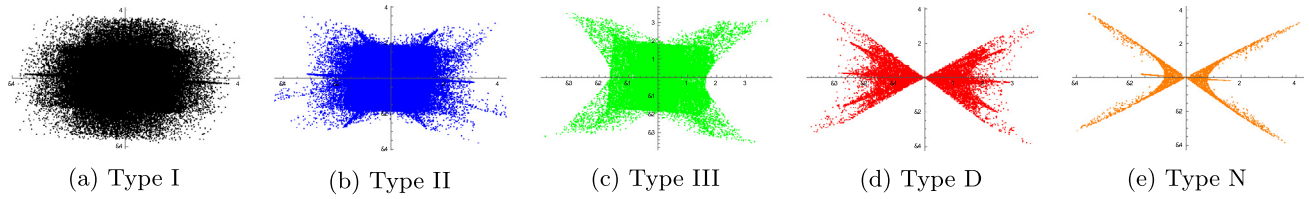
5. Outlook

In this work we have shown how to apply techniques from machine learning and data science in classification problems in general relativity. Taking as an example the Petrov classification of the Weyl tensor, we have adapted the problem to fit into the realm of supervised machine learning.

That is, our input consisted of randomly generated five-dimensional vectors representing the Weyl scalars $\{\Psi_i\}$ $(i = 0, ..., 4)$, labelled with their corresponding Petrov type (I, II, III, D, N, O). We generated enough data points to consider all possible cases of non-vanishing Weyl components, as described by Table 1, to have a

[1] Here and in the following we will use the dataset built from real entries for Ψ_i. An analog analysis was produced for the complex dataset, where the input vector is ten-dimensional, after splitting in real and imaginary parts. Similar results in accuracy and confidence were found for the complex dataset.

[2] Alternatively, we can also compute the F_1-score for each class and then the weighted F_1-score for the whole dataset. In our calculations, we find the F_1-score to be 0.979.



(a) Type I (b) Type II (c) Type III (d) Type D (e) Type N

Fig. 4. Visual representation of the different Petrov types. The data was dimensionally reduced using Principal Component Analysis (PCA) to observe the directions with the highest variance.

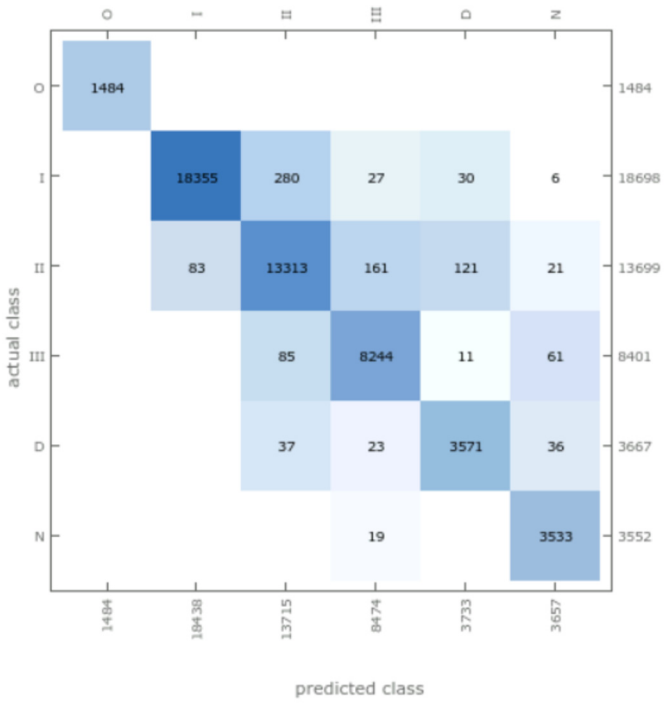


Fig. 5. A plot of the Confusion Matrix by our NN classifier; we can see that it is heavily diagonal, signifying that the classification into the 6 Petrov types is extremely accurate.

set of base-independent training data. We designed a feed-forward neural network to train on this data and achieved 98% accuracy with a confidence (MCC) of 0.973. This shows that with a very simple neural network, in only a handful of epochs, one can model the Petrov classification with a high degree of success. We also showed how data visualization and dimensionality reduction can help in analyzing the data itself and the patterns that underlie it. Both these directions can help illuminate the intricacies of the classification of spacetimes, shedding light on problems of numerical relativity or in the general study of solutions to the Einstein equations.

The Petrov classification is only a part of the general programme for classifying and comparing spacetimes. The procedure elucidated on this paper can easily be extended to model the Segre classification of the Ricci tensor, for another part of the puzzle. This then constitutes the first step in having a machine ready setup for the full classification of spacetimes, a machine learning formulation of the Cartan-Karlhede algorithm. With the development of online databases for exact solutions to the Einstein equations, the stage is set for a complete exploration of the power of these techniques in this important field.

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 for grant ST/J00037X/2.

Appendix A. The Petrov-Penrose classification of spacetimes

In this appendix we provide the conventions and mathematical background used to define Petrov's classification, basing our analysis on [1]. This also sets the notation for the main text, especially Fig. 1.

A complex null tetrad is a choice of two real null vectors $\mathbf{l}, \mathbf{k}$ and two complex conjugate null vectors $\mathbf{m}, \bar{\mathbf{m}}$:

$$\mathbf{e}_a = (\mathbf{m}, \bar{\mathbf{m}}, \mathbf{l}, \mathbf{k}) , \tag{A.1}$$

with the only non-vanishing products

$$k^a l_a = -1 , \qquad m^a \bar{m}_a = 1 , \tag{A.2}$$

and where the metric in this basis reads

$$g_{ab} = 2m_{(a}\bar{m}_{b)} - 2k_{(a}l_{b)} . \tag{A.3}$$

From this tetrad we can build a basis of bivectors with components

$$\begin{aligned} U_{ab} &= -l_a\bar{m}_b + l_b\bar{m}_a , \\ V_{ab} &= k_a m_b - k_b m_a , \\ W_{ab} &= m_a\bar{m}_b - m_b\bar{m}_a - k_a l_b + k_b l_a , \end{aligned} \tag{A.4}$$

that will be useful in the following.

We remember that the Weyl tensor is the trace-free part of the curvature tensor, given by

$$\begin{aligned} C_{abcd} =& R_{abcd} + \frac{1}{2}(R_{bc}g_{ad} + R_{ad}g_{bc} - R_{bd}g_{ac} - R_{ac}g_{bd}) \\ &+ \frac{1}{6}R(g_{ac}g_{bd} - g_{ad}g_{bc}) . \end{aligned} \tag{A.5}$$

This tensor has the same symmetries as the Riemann curvature, with the added property of tracelessness. In general, it has ten independent components.

For the classification it is useful to define the complex tensor

$$C^*_{abcd} \equiv C_{abcd} + \mathrm{i}\tilde{C}_{abcd} \tag{A.6}$$

where

$$\tilde{C}_{abcd} \equiv \frac{1}{2}\varepsilon_{cdef}C^{ef}_{ab} . \tag{A.7}$$

Now we can expand C^*_{abcd} in the basis (A.4) as

$$\frac{1}{2}C^*_{abcd} = \Psi_0 U_{ab}U_{cd} + \Psi_1(U_{ab}W_{cd} + W_{ab}U_{cd}) + \Psi_2(V_{ab}U_{cd} + U_{ab}V_{cd} + W_{ab}W_{cd}) + \Psi_3(V_{ab}W_{cd} + W_{ab}V_{cd}) + \Psi_4 V_{ab}V_{cd}\,, \tag{A.8}$$

with the five complex coefficients defined by

$$\begin{aligned}
\Psi_0 &\equiv C_{abcd}k^a m^b k^c m^d\,,\\
\Psi_1 &\equiv C_{abcd}k^a l^b k^c m^d\,,\\
\Psi_2 &\equiv C_{abcd}k^a m^b \bar{m}^c l^d\,,\\
\Psi_3 &\equiv C_{abcd}k^a l^b \bar{m}^c l^d\,,\\
\Psi_4 &\equiv C_{abcd}\bar{m}^a l^b \bar{m}^c l^d\,.
\end{aligned} \tag{A.9}$$

Therefore, determining the ten independent components of the Weyl tensor in (A.5) is equivalent to determining the five complex scalars defined above. With regards to their physical interpretation: Ψ_0 and Ψ_1 represent transverse and longitudinal waves in the $\mathbf{l}$ direction, Ψ_2 a Coulomb-like component and Ψ_3 and Ψ_4 are longitudinal and transverse wave components in the $\mathbf{k}$ direction.

Petrov's classification by Penrose [31] characterizes the Weyl tensor according to principal null directions $\mathbf{k}$ with the property

$$k_{[e}C_{a]bc[d}k_{f]}k^b k^c = 0 \tag{A.10}$$

There can be at most four such null vectors (p.n.d.'s). If a spacetime admits four distinct p.n.d.'s it is called algebraically general (type I), otherwise it is algebraically special.

If $\mathbf{k}$ is a member of the null tetrad then equation (A.10) is equivalent to $\Psi_0 = 0$. We can rotate to an arbitrary complex null tetrad $(\mathbf{m}', \bar{\mathbf{m}}', \mathbf{l}', \mathbf{k}')$, where the coefficient Ψ_0 undergoes the transformation:

$$\Psi_0 = \Psi'_0 - 4z\psi'_1 + 6z^2\Psi'_2 - 4z^3\Psi'_3 + z^4\Psi'_4\,, \tag{A.11}$$

with z a complex number. So we see that the determination of principal null directions is equivalent to solving the quartic equation for z:

$$\Psi'_0 - 4z\psi'_1 + 6z^2\Psi'_2 - 4z^3\Psi'_3 + z^4\Psi'_4 = 0\,, \tag{A.12}$$

showing that there can be indeed four (complex) roots to this equation, that do not need to be different. Depending on the amount and multiplicity of the p.n.d.'s we get the classification in Fig. 1.

References

[1] H. Stephani, D. Kramer, M.A.H. MacCallum, C. Hoenselaers, E. Herlt, Exact Solutions of Einstein's Field Equations, Cambridge Monographs on Mathematical Physics., Cambridge Univ. Press, Cambridge, 2003.

[2] M.A.H. MacCallum, Computer algebra in gravity research, Living Rev. Relativ. 21 (1) (2018) 6.

[3] A. Karlhede, A review of the geometrical equivalence of metrics in general relativity, Gen. Relativ. Gravit. 12 (1980) 693–707.

[4] A.Z. Petrov, Klassifikacya prostranstv opredelyayushchikh polya tyagoteniya, Uch. Zapiski Kazan. Gos. Univ. 114 (1954) 55–69.

[5] C. Segre, Sulla teoria e sulla classificazione delle omografie in uno spazio lineare ad uno numero qualunque di dimensioni, Mem. R. Accad. Lincei 3a (1884) 127.

[6] D. Pollney, J.E.F. Skea, R.A. d'Inverno, Classifying geometries in general relativity: III. Classification in practice, Class. Quantum Gravity 17 (15) (Aug. 2000) 2885–2902.

[7] Y.-H. He, Deep-learning the landscape, 6, arXiv:1706.02714 [hep-th], 2017, q.v. Interview in Science 365 (July 2019).

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

[9] D. Krefl, R.-K. Seong, Machine learning of Calabi-Yau volumes, Phys. Rev. D 96 (6) (2017) 066014, arXiv:1706.03346 [hep-th].

[10] J. Carifio, J. Halverson, D. Krioukov, B.D. Nelson, Machine learning in the string landscape, J. High Energy Phys. 09 (2017) 157, arXiv:1707.00655 [hep-th].

[11] F. Ruehle, Evolving neural networks with genetic algorithms to study the String Landscape, J. High Energy Phys. 08 (2017) 038, arXiv:1706.07024 [hep-th].

[12] Y.-H. He, M. Kim, Learning algebraic structures: preliminary investigations, arXiv:1905.02263 [cs.LG].

[13] L. Alessandretti, A. Baronchelli, Y.-H. He, Machine learning meets number theory: the data science of Birch-swinnerton-dyer, arXiv:1911.02008 [math.NT].

[14] Y.-H. He, S.-T. Yau, Graph Laplacians, Riemannian manifolds and their machine-learning, arXiv:2006.16619 [math.CO].

[15] Y.-H. He, Machine-learning mathematical structures, arXiv:2101.06317 [cs.LG].

[16] 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 (7887) (2021) 70–74.

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

[18] F. Ruehle, Data science applications to string theory, Phys. Rep. 839 (2020) 1–117.

[19] S. Krippendorf, M. Syvaeri, Detecting symmetries with neural networks, arXiv:2003.13679 [physics.comp-ph].

[20] C. Krishnan, V. Mohan, S. Ray, Machine learning $\mathcal{N} = 8$, $D = 5$ gauged supergravity, Fortschr. Phys. 68 (5) (2020) 2000027, arXiv:2002.12927 [hep-th].

[21] H.-Y. Chen, Y.-H. He, S. Lal, M.Z. Zaz, Machine learning etudes in conformal field theories, arXiv:2006.16114 [hep-th].

[22] Z. Liu, M. Tegmark, Machine-learning hidden symmetries, arXiv:2109.09721 [cs.LG].

[23] S. Alexander, W.J. Cunningham, J. Lanier, L. Smolin, S. Stanojevic, M.W. Toomey, D. Wecker, The autodidactic universe, arXiv:2104.03902 [hep-th].

[24] R. Altman, J. Carifio, X. Gao, B. Nelson, Orientifold Calabi-Yau threefolds with divisor involutions and string landscape, arXiv:2111.03078 [hep-th].

[25] X. Gao, H. Zou, Machine learning to the orientifold Calabi-Yau with string vacua, arXiv:2112.04950 [hep-th].

[26] R.A. d'Inverno, R.A. Russell-Clark, Classification of the harrison metrics, J. Math. Phys. 12 (1971) 1258–1263.

[27] F.W. Letniowski, R.G. McLenaghan, An improved algorithm for quartic equation classification and petrov classification, Gen. Relativ. Gravit. 20 (1988) 463–483.

[28] J.E. Aman, R.A. D'Inverno, G.C. Joly, M.A.H. MacCallum, Quartic equations and classification of Riemann tensors in general relativity, Gen. Relativ. Gravit. 23 (9) (Sept. 1991) 1023–1055.

[29] E. Zakhary, K. Vu, J. Carminati, A new algorithm for the petrov classification of the weyl tensor, Gen. Relativ. Gravit. 35 (07 2003) 1223–1242.

[30] E. Newman, R. Penrose, An approach to gravitational radiation by a method of spin coefficients, J. Math. Phys. 3 (3) (1962) 566–578.

[31] R. Penrose, A spinor approach to general relativity, Ann. Phys. 10 (2) (1960) 171–201.

[32] I.V. Tanatarov, O.B. Zaslavskii, What happens to Petrov classification on horizons of axisymmetric dirty black holes, J. Math. Phys. 55 (2014) 022502, arXiv:1211.4376 [gr-qc].

[33] W.R. Inc., Mathematica, version 13.0.0, https://www.wolfram.com/mathematica, 2021, Champaign, IL.

[34] D.P. Kingma, J. Ba, Adam: A Method for Stochastic Optimization, 2017.