

Graph Laplacians, Riemannian Manifolds & their Machine-Learning

Yang-Hui He^{1,2,3} & Shing-Tung Yau^{4,5,6}

- ¹ *Merton College, University of Oxford, OX14JD, UK*
² *Department of Mathematics, City, University of London, EC1V 0HB, UK*
³ *School of Physics, NanKai University, Tianjin, 300071, P.R. China*
⁴ *Department of Mathematics, Department of Physics, & Center of Mathematical Sciences and Applications, Harvard University, Cambridge, MA 02138, USA*
⁵ *Yau Mathematical Sciences Center, Tsinghua University, Beijing, 100804, China*
⁶ *Beijing Institute of Mathematical Sciences and Applications, Huairou Science City, Beijing, 101400, China*

hey@maths.ox.ac.uk; yau@math.harvard.edu

Abstract

Graph Laplacians as well as related spectral inequalities and (co-)homology provide a foray into discrete analogues of Riemannian manifolds, providing a rich interplay between combinatorics, geometry and theoretical physics. We apply some of the latest techniques in data science such as supervised and unsupervised machine-learning and topological data analysis to the Wolfram database of some 8000 finite graphs in light of studying these correspondences. Encouragingly, we find that neural classifiers, regressors and networks can perform, with high efficiency and accuracy, a multitude of tasks ranging from recognizing graph Ricci-flatness, to predicting the spectral gap, to detecting the presence of Hamiltonian cycles, etc.

Contents

1	Introduction and Summary	2
2	Graph Laplacians	4
2.1	Rudiments on the Laplacian	5
2.2	Connection to Geometry	7
3	The Wolfram Database of Simple Graphs	9
3.1	Preliminary Machine-Learning of Graph Properties	10
3.1.1	A Graphical Miscellany	14
3.1.2	Maximal Laplacian Eigenvalue	18
3.1.3	Spectral Gap	21
4	Laplacian Spectra and Inequalities	22
4.1	Eigenvalue Distributions of the Normalized Laplacian	23
4.2	Machine-Learning Spectral Bounds	23
4.2.1	Unsupervised Treatments	24
4.2.2	Supervised Learning	25
5	Ricci Flat Graphs	27
5.1	Two Notions of Ricci-Flatness for Graphs	27
5.1.1	Classification Results	29

5.2	Distinguishing OLLY-Ricci-Flat Graphs with ML	31
5.3	Distinguishing CD-Ricci-Flat Graphs with ML	32
6	Homology and Cohomology of Graphs	33
6.1	Machine Learning Graph Euler Number	36
7	Conclusions and Prospectus	37
A	Illustrative Example	40

1 Introduction and Summary

The Laplacian is of paramount importance in mathematics, its ubiquity has extended from curvature in differential geometry, to (co-)homology in algebraic geometry, to invariants in topology and to particle spectrum in high energy physics. In classical Riemannian geometry, the celebrated decomposition of Hodge relates the zero-modes of the Laplacian to the representative in cohomology, and hence allows for an elegant computation H^* . In complex geometry, the second author’s reduction of the Laplacian eigen-equation for Kähler manifolds to analyses of a PDE of Monge-Ampère type [Yau] resulted in the proof of Calabi’s Conjecture [Ca].

A natural question arose as to whether there should exist a *discrete* version of the story. A programme had been launched over the last decade or so to understand it in the context of locally finite graphs. Based on the standard theory of the Laplace operator on graphs (cf. e.g., [Fan]), the authors in [FanYau, LLY, LY2, HuL, GLLY, CK-LLLY] pursued a notion of curvature on graphs to examine the analogue for manifolds (q.v. a comprehensive review in [LY]). Furthermore, [GLMY1, GLMY2] investigated the idea of (co-)homology of finite graphs.

In parallel, a recent programme of using latest techniques from machine-learning and data science to study various mathematical formulae and conjectures had been proposed [He, HeBook]. Experimentation of whether standard techniques in neural

networks and classifiers could be carried over to study diverse problems in mathematics have ranged from triangulations in Calabi-Yau hypersurfaces in toric varieties [ACHN,DLMS,HJP], to finding bundle cohomology on varieties [Rue,BCDL,LS], to distinguishing elliptic fibrations [AGGL,HL] and invariants of Calabi-Yau threefolds [BHJM] (cf. [GRV,GH] on the organization of classification by Hasse diagrams), to knot hyperbolic volumes [JKP], to machine-learning the Donaldson algorithm for numerical Calabi-Yau metrics [AHO], to the algebraic structures of groups and rings [HK], to the BSD conjecture in number theory [ABH], to finding discriminant locii [BHMRT] etc. (q.v. [HeTalk] for speculations on how the foundations of mathematics might respond to machine-learning).

Indeed, with the introduction of the machine-learning paradigm [He,KS,Rue,CHKN] to string theory, a multitude of heartening results in physics have included, e.g., finding Higgsable gauge groups [WZ], axion physics from strings [DLMS], flux compactifications [CSS], distinguishing standard models [MPV,OT,DHLL], QFT dualities [BK,BFHHM,HHP], detecting symmetries [KSy] and CFTs [CHLZ], etc. Of particular note is [HMT] where the holographic AdS/CFT correspondence and thence, space-time itself, is interpreted as a neural-network/Boltzmann machine. The reader is furthermore referred to fascinating works in the last couple of years on neural-networks which can perform symbolic mathematics [LC], find results in fundamental physics [IMWDR,CSBXCSH] from scratch and discover chemical reactions [Tsh] from word-embedding (cf. a similar linguistic study [HJN] of ArXiv titles in classifying different disciplines in mathematical physics and in generating syntactical identities).

Given the highly structured representation of finite graphs in terms of matrix manipulations - something for which machine-learning is perfectly adapted - it is immediate to ask whether the aforementioned two programmes should intersect. That is, whether machine-learning could be applied to the investigation of the geometry of graphs. It is this question which we will address in this paper, which will hopefully initiate a novel direction in the study of the geometry and algebra in graph theory, especially in the context of discrete analogues to Riemannian manifolds.

Bearing this question in mind, the paper is organized as follows. In Section 2, we begin with some rudiments on the graph Laplacian and related standard facts in graph theory, as well as the connection to Riemannian geometry. Then, in Section 3, we present our main protagonist of the Wolfram Database of finite, undirected, simple graphs, a set of some 8000 graphs organized by vertices and edges, and named

wherever possible. On this set we examine some preliminary properties, such as planarity, genus, chromatic number, etc., to see whether they can be machine-learned.

Section 4 is then concerned with spectral bounds. These are bounds on the eigenvalues of the Laplacian, which are not only important to the classical spectral theory of graphs, but are also enlightening to Li-Yau type of inequalities [LiYau] in compact manifolds. We study the distribution of the eigenvalues and then apply topological data analysis, principal component analysis, as well as supervised machine-learning to these bounds. Section 5 is devoted to the critical class of finite graphs which are Ricci-flat. These graphs are the analogues of Calabi-Yau manifolds, and the classification thereof has been an active field of research [LLY, LY2, HuL]. We will show that a neural classifier can distinguish a Ricci-flat graph to very high accuracy.

Finally, Section 6 is devoted to studying the homology of directed (finite, simple) graphs. We explicitly compute the Euler number of the graphs in the sense of [GLMY1, GLMY2] for the Wolfram database and see how machine-learning “guesses” at the answer. We conclude with prospects and outlook in Section 7.

2 Graph Laplacians

We commence with some rudiments from the theory of graphs, especially in relation to the Laplacian and its analogue for manifolds. A graph G is a pair (V, E) where V is a set of **vertices** (we will also refer to them as nodes interchangeably) and E , a set of **edges** (or arrows, whenever the edges are directed). V could be infinite but in this paper we will consider only finite graphs where the vertices in V can be labeled as $v_{i=1, \dots, n}$. A directed arrow $i \rightarrow j$ links v_i to v_j and a self-adjointing arrow $i \rightarrow i$ is called a loop. A path in G is a sequence of arrows $\{v_{i_0} \rightarrow v_{i_1}, v_{i_1} \rightarrow v_{i_2}, v_{i_2} \rightarrow v_{i_3}, \dots, v_{i_k} \rightarrow v_{i_{k+1}}\}$. If the start and ending points of a path is the same node, i.e., $v_{i_0} = v_{i_{k+1}}$, then the path is called a **cycle**.

As defined, we are allowing for multiple arrows between nodes. When we disallow (1) multiple edges between nodes and (2) loops which link a node to itself, then G is called **simple**. Note that simple graphs *do allow* for cycles. Furthermore, if we ignore orientation of all arrows (and refer to them simply as edges), we have undirected graphs. In congruence with the data available to us, which we will discuss shortly,

we henceforth focus on finite, simple, undirected graphs, unless otherwise stated.

We now define some standard concepts in the theory of graphs.

DEFINITION 1. The **adjacency matrix** of $G = (V, E)$ with $|V| = n$ vertices is an

$$n \times n \text{ matrix } A_{ij} = \begin{cases} 1, & \text{if edge } i \rightarrow j \in E \\ 0, & \text{otherwise.} \end{cases}$$

We use \sim for adjacency: $v_i \sim v_j$ means nodes i and j is linked by an edge $i \rightarrow j$. The **degree** of vertex v_i is the number of its neighbours: $d(v_i) = \sum_{v_i \sim v_j} 1$. We record the degrees into a diagonal matrix, the **degree matrix** D_{ii} whose i -th diagonal entry is the degree of vertex v_i .

In the case of our undirected graphs, A is symmetric. Moreover, for simple graphs the diagonal entries are all 0 (no loops) and all non-zero entries are 1 (no multi-edges).

Thus prepared, we can define the graph Laplacian simply as

DEFINITION 2. For graph $G = (V, E)$, the Laplacian is defined as

$$L := D - A ; \quad L_{ij} = \begin{cases} d(v_i) & i = j \\ -1 & i \neq j \text{ and } v_i \sim v_j \\ 0 & \text{otherwise.} \end{cases}$$

We remark that for our simple undirected graphs with n nodes, this is a symmetric $n \times n$ matrix, with diagonal entries being the degrees of the nodes.

2.1 Rudiments on the Laplacian

As defined above, it may seem obscure as to why L is called the Laplacian. To facilitate our grasp, we require several concepts:

DEFINITION 3. The **incidence matrix** ∇ of $G = (V, E)$ is an $|E| \times |V|$ matrix with subscript $i = 1, \dots, |V|$ and $e = 1, \dots, |E|$, such that for each edge $i \rightarrow j$, $\nabla_{ie} = -1$, $\nabla_{je} = 1$, and 0 otherwise. In other words, $(-1, +1)$ records the initial and final vertex of each edge, indexed by the rows.

For undirected graphs, we can choose some arbitrary but fixed orientation and whereby define ∇ .

Next, we can define a (real-valued) function $f : G = (V, E) \rightarrow \mathbb{R}$ on a graph by assigning a real value to each node as $f(v_i)$. We denote the space of such functions as

$$V^{\mathbb{R}} := \{f : V \rightarrow \mathbb{R}\} . \quad (2.1)$$

Subsequently, the adjacency and Laplacian matrices of G can be seen as operators on functions on G . For example, the adjacency matrix can be written as

$$A : f \longrightarrow g := A(f) ; g(v_i) = \sum_{i \sim j} f(v_j) . \quad (2.2)$$

For our undirected graphs, this gives a convenient symmetric quadratic form $f^T A f = \sum_{e \in E} f(v_i) f(v_j)$. To illustrate, the following is a 4-vertex, 5-edged simple graph (the diamond graph), for which we write the adjacency matrix, choose an orientation, and write the incidence matrix:

$$A = \begin{pmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix} ,$$

$$D = \begin{pmatrix} 2 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 3 \end{pmatrix}$$

$$\nabla = \begin{pmatrix} -1 & 0 & 1 & 0 \\ -1 & 0 & 0 & 1 \\ 0 & 0 & 1 & -1 \\ 0 & 1 & 0 & -1 \\ 0 & 1 & -1 & 0 \end{pmatrix}$$

(2.3)

As the notation suggests, ∇ is a “differential” operator on functions on G : it is a co-boundary map in the sense that on an edge $i \rightarrow j$, $(\nabla f)(i \rightarrow j) = f(v_i) - f(v_j)$. In analogy, we have

PROPOSITION 1. *The graph Laplacian is the “square” of the incidence matrix*

$$L = D - A = \nabla^T \nabla ; \quad (Lf)(v_i) = \sum_{v_i \sim v_j} f(v_i) - f(v_j) ,$$

irrespective of the choice of orientation.

Sketch Proof: To see this, we simply observe that $\sum_e \nabla_{ie} \nabla e i$ contributes 1 to each edge incident upon node i , which is then summed over the edges, i.e., it gives the degree of node i . Similarly, $\sum_e \nabla_{ie} \nabla e j$ for $i \neq j$ contributes $-1 \cdot 1 = -1$ to the (i, j) -entry of L , which is the negative of the adjacency matrix. Moreover, changing the direction of any edge in the direction assignment does not change the value of $\nabla_{ie} \nabla e j$ because one will be $+1$ and the other -1 , whose product remains -1 .

As we consider only undirected graphs, the relevant matrices A and L are symmetric, whereby giving us only real eigenvalues, these are respectively called the adjacency **spectrum** and Laplacian spectrum of the graph. Moreover, an important corollary of Proposition 1 is that not only is the Laplacian spectrum real, it is also non-negative:

$$\lambda_i = \text{Eigenvalues}(L) ; \quad 0 \leq \lambda_1 \leq \lambda_2 \leq \dots \leq \lambda_{n=|V|} . \quad (2.4)$$

This is seen by considering the eigensystem $L\phi = \lambda\phi$, which can be rewritten as $\lambda = \phi^T L \phi = \phi^T \nabla^T \nabla \phi = |\nabla \phi|^2 \geq 0$.

Finally, we remark that oftentimes we *weight* the edges of the graph by assigning a real positive number (the weight) w_{ij} to each edge $i \rightarrow j$. Calling the set of weights W , we consider undirected weighted graphs $G = (V, E, W)$ and modify all of the above concepts accordingly. The weighted adjacency matrix A of $G = (V, E, W)$ is such that $A_{ij} = w_{ij}$ and 0 otherwise. Likewise, the weighted Laplacian is $L = D - A ; (Lf)(v_i) = \sum_{v_j \sim v_i} w_{ij}(f(v_i) - f(v_j))$.

2.2 Connection to Geometry

The foregoing definitions and results are standard and can be found, for example, in [Fan]. In the ensuing we will adhere to the discussions and conventions of [LY], in light of connections to differential and algebraic geometry. First, we re-scale the Laplacian as

$$\Delta := D^{-1}L = I - D^{-1}A ; \quad (\Delta f)(v_i) = \frac{1}{d(v_i)} \sum_{v_i \sim v_j} f(v_i) - f(v_j) . \quad (2.5)$$

In the literature, this is often called the **random walk normalized graph Laplacian** and is the one used in the programme of Yau et al. Likewise, we define a normalized version of the incidence (coboundary map):

$$|\nabla f(v_i)|^2 := \frac{1}{d(v_i)} \sum_{v_i \sim v_j} (f(v_i) - f(v_j))^2 \quad (2.6)$$

We make an important remark that in [LY] (q.v., the first definition of Δ on p2), the Laplacian therein actually is the negative of (2.5), which differs from some of the graph-theory literature. However, this negative sign is compensated by its re-insertion in the eigen-equation on p4. We will forego this double-negative and adhere to (2.5).

Now, upon defining a bilinear operator $V^{\mathbb{R}} \times V^{\mathbb{R}} \rightarrow V^{\mathbb{R}}$ as

$$\Gamma(f, g)(x) = \frac{1}{2}(\Delta(fg) - f\Delta g - g\Delta f) , \quad (2.7)$$

the concept of the curvature of a graph was introduced by [FanYau]:

DEFINITION 4. *The Ricci curvature of a graph is given by*

$$\Gamma_2(f, g)(x) = \frac{1}{2}(\Delta\Gamma(fg) - \Gamma(f, \Delta g) - \Gamma(g, \Delta f))(x) .$$

We now recall from differential geometry that for a compact, smooth and complete Riemannian manifold M , there is **Bochner's formula** which relates harmonic functions $u : \Delta u = 0$ on M to the Ricci curvature Ric (cf. [CLN]):

$$\frac{1}{2}\Delta(|\nabla u|^2) = |\nabla^2 u|^2 + Ric(\nabla u, \nabla u) , \quad (2.8)$$

where ∇u is the gradient of u and Ric , the Ricci curvature tensor, both with respect to the Riemannian metric on M .

One of the initial motivations of [LY, LY2] was to have a discrete, graph-theoretic, versions of the above. In particular, one has that [LY2] (cf. also [GLLY])

THEOREM 1 (Lin-Yau). *Let $G = (V, E)$ be a locally ¹ finite graph and $d(G) :=$*

¹ Note that the conditions here are more general than what we need and the graph itself can have an infinite number of vertices. Locally finite means that at least all vertex degrees are finite. The supremum over all such degrees can, however, be infinite; hence we take the sup rather than max.

$\sup_{x \in V} d_x$ is the supremum over all vertex degrees (it can be that $d(G) = \infty$), then

$$\Gamma_2(f, f) \geq \frac{1}{2}(\Delta f)^2 + \left(\frac{1}{d(G)} - 1 \right) \Gamma(f, f) ,$$

for any $f \in V^{\mathbb{R}}$.

We will return to address such inequalities in §4.

3 The Wolfram Database of Simple Graphs

As a concrete play-ground, we take the graph database from Wolfram [Graph], as implemented in Mathematica [Wolf], up to 100 vertices. This is a list of undirected simple graphs, totaling 7785, which is a sizable set upon which we shall experiment. We emphasize that this number is far less than the total number of known non-isomorphic, connected simple graphs up to 100 nodes, which proceeds exponentially [OEISg] with the number of nodes as

$$1, 1, 1, 2, 6, 21, 112, 853, 11117, 261080, 11716571, 1006700565, 164059830476, \dots \quad (3.1)$$

$$50335907869219, 29003487462848061, \dots$$

A histogram of the number of vertices (dimension of adjacency matrix) of the list of graphs is drawn in Part (a1) of Figure 2

As explicit examples, we have Octahedral graph of 6 vertices in part (a) and the complete bipartite of 9 vertices in part (b) of Figure 1. For reference, the Laplacians

are $L_{\text{Octa}} = \begin{pmatrix} 4 & -1 & -1 & -1 & -1 & 0 \\ -1 & 4 & -1 & -1 & 0 & -1 \\ -1 & -1 & 4 & 0 & -1 & -1 \\ -1 & -1 & 0 & 4 & -1 & -1 \\ -1 & 0 & -1 & -1 & 4 & -1 \\ 0 & -1 & -1 & -1 & -1 & 4 \end{pmatrix}$ and $L_{\text{Bipartite}(5,4)} = \begin{pmatrix} 5 & 0 & 0 & 0 & -1 & -1 & -1 & -1 & -1 \\ 0 & 5 & 0 & 0 & -1 & -1 & -1 & -1 & -1 \\ 0 & 0 & 5 & 0 & -1 & -1 & -1 & -1 & -1 \\ 0 & 0 & 0 & 5 & -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & 4 & 0 & 0 & 0 & 0 \\ -1 & -1 & -1 & -1 & 0 & 4 & 0 & 0 & 0 \\ -1 & -1 & -1 & -1 & 0 & 0 & 4 & 0 & 0 \\ -1 & -1 & -1 & -1 & 0 & 0 & 0 & 4 & 0 \\ -1 & -1 & -1 & -1 & 0 & 0 & 0 & 0 & 4 \end{pmatrix}$. We

include in the Appendix a detailed walk-through of a particular example, exemplifying all the concepts encountered in the main body.

As mentioned in (2.4), all eigenvalues of the Laplacians of undirected graphs are non-negative. To give an idea of the distribution of the spectrum, we show the histogram in part (a) of Figure 2. In part (b) of the figure, we find that a good fit is

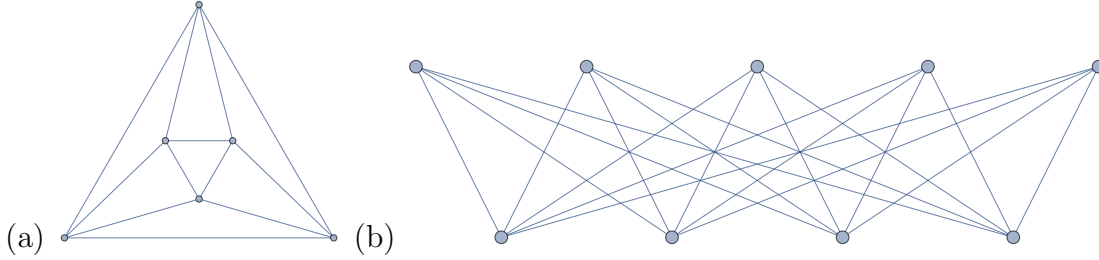


Figure 1: (a) the Octahedral graph of 6 nodes; (b) the complete bipartite graph of 9 nodes from Wolfram GraphData[] database.

a Weibull distribution

$$p(x) = (x - \mu)^{\alpha-1} e^{-\left(\frac{x-\mu}{\beta}\right)^\alpha} \Theta(x - \mu) ; \quad \alpha = 1.16, \beta = 5.20, \mu = -0.05 , \quad (3.2)$$

where $\Theta(x)$ is the step function which is 1 for $x \geq 0$ and 0 for $x < 0$. These should be compared with theoretical results on spectrum distributions of random graphs in [GM,DJ] as well as that of the Laplacian eigenvalue distribution [DMT].

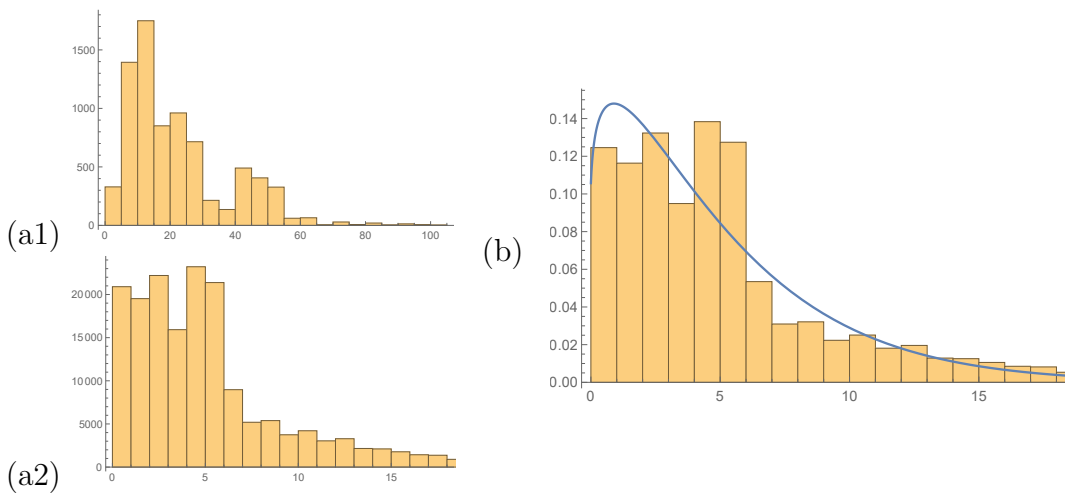


Figure 2: (a1) The histogram of the number of vertices of the Wolfram database of 7785 undirected simple graphs up to 100 nodes; (a2) The histogram of their Laplacian eigenvalues; (b) The associated probability density (normalized histogram) and a fit by the Weibull distribution.

3.1 Preliminary Machine-Learning of Graph Properties

With our database it is expedient that we test whether some preliminary key graph-theoretical quantities can be machine-learned. This is in the spirit of the citations

in the introduction of whether various relevant quantities in string theory and mathematical physics can be deep-learnt. In particular, the question of whether fundamental mathematical structures can be detected by AI in a supervised way has been pursued in the programme of [He, HL, HK, ABH, HHP], ranging from number theory to representation theory, to computational geometry. Importantly, the idea is to use neural networks and classifiers that have *no prior knowledge* of the mathematics: indeed, with specifically chosen architectures guided by human intuition one achieves high precision (and even generate exact formulae as in [BCDL]), but can generic networks find unseen patterns? This seems to be the case in many of the aforementioned instances.

Our paradigm is clear. Take our dataset \mathcal{D} of the Wolfram graphs. Many important quantities have been computed and compiled. Take property \mathcal{P} and split $\mathcal{D} = \mathcal{T} \sqcup \mathcal{V}$, the training set and validation set. The training set then consists of association rules (labels) of the form $\mathcal{T} = \{A_i \rightarrow \mathcal{P}_i\}$ with adjacency matrix A_i for the i -th graph is to be associated with property \mathcal{P}_i ; an appropriate neural network or similar machine-learning algorithm is then trained with \mathcal{T} . The result is then validated against \mathcal{V} which the machine has *not* seen before in order to avoid over-fitting. The validation is done, in the same spirit as a regression, by letting the algorithm *predict* the property \mathcal{P} for elements in \mathcal{V} , after which we can compute some measure of “goodness of fit”. What we have described above is the paradigm of **supervised learning**, where clear properties (labels) are assigned to the training set.

Typically, if \mathcal{P} is continuous, one can find a linear regression on the predicted versus actual values of \mathcal{P} on the validation set \mathcal{V} , the fit should be as close to the line $y = x$ as possible. If \mathcal{P} is discrete (and of a finite set of values, which can be labeled as $1, 2, \dots, k$), this is called “ k -categorical data” (when $k = 2$, this is binary classification). Here, a **confusion matrix** C_{ij} may be set up, where i indexes the actual k categories and j indexes the machine-predicted k -categories so that 1 is added to each such pair. In the ideal case of perfect prediction, C_{ij} is a diagonal matrix.

Several standard measure of the quality of the prediction (q.v. e.g., [She]). There is the naive **precision** which is the percentage of correct predictions:

$$P_N := \sum_i C_{ii} / \sum_{i,j} C_{ij} . \tag{3.3}$$

The naive precision is too crude in that it does not distinguish amongst the diagonal entries. A good measure (cf. [Matt]) for k -categorical classification is the **Matthews correlation coefficient**:

$$\phi := \frac{\sum_k \sum_l \sum_m C_{kk} C_{lm} - C_{kl} C_{mk}}{\sqrt{\sum_k (\sum_l C_{kl}) (\sum_{k'|k' \neq k} \sum_{l'} C_{k'l'})} \sqrt{\sum_k (\sum_l C_{lk}) (\sum_{k'|k' \neq k} \sum_{l'} C_{l'k'})}} \quad (3.4)$$

The value of $\phi \in [-1, 1]$ where 1 means perfect matching, 0 means random correlation and -1 means anti-correlation.

In the case of binary classifications where there are only 2 categories, the confusion matrix is usually denoted as

$$C = \begin{array}{c} \begin{array}{cc|cc} & & \text{Actual} & \\ & & \text{True (1)} & \text{False (0)} \\ \text{Predicted} & \text{True (1)} & \text{True Positive (} tp \text{)} & \text{False Positive (} fp \text{)} \\ \text{Classification} & \text{False (0)} & \text{False Negative (} fn \text{)} & \text{True Negative (} tn \text{)} \end{array} \end{array} \quad (3.5)$$

and the Matthews correlation coefficient reduces

$$\phi = \frac{tp \cdot tn - fp \cdot fn}{\sqrt{(tp + fp)(tp + fn)(tn + fp)(tn + fn)}}. \quad (3.6)$$

In such binary classifications, another measure is the F1-score, which also needs to be close to 1 for a good prediction:

$$F_1 := \frac{2}{\frac{1}{TPR} + \frac{1}{Precision}}, \quad \text{TPR} := \frac{tp}{tp+fn}, \quad \text{FPR} := \frac{fp}{fp+tn}, \quad (3.7)$$

$$\text{Accuracy } p := \frac{tp+tn}{tp+tn+fp+fn}, \quad \text{Precision} := \frac{tp}{tp+fp}.$$

where TPR (FPR) stands for true (false) positive rate.

Armed with the rudiments, we can immediately turn to some important properties as warm-up exercises to our machine-learning perspective on graphs. Now, the intersection of graph theory and machine-learning is taking shape recently (q.v., e.g., [Coh, Nik] and references therein). However, we point out that our approach is paradigmatically different here. We are not employing results from graph theory to establish and study neural networks. Instead, we are doing the opposite of us-

ing standard ML technique to see whether crucial graph properties can be learnt by supervision. This is in line with the programme of the first author over the last two years in trying to see whether mathematical structures can be machine-learned (cf. summary talk at [HeTalk]).

In the following, we will perform what is called **5-fold cross-validation**, where we split (sequentially is good enough) \mathcal{D} into 5 groups, train on a subgroup of these and validate against the remaining. In particular, we will show performance with

$$\begin{aligned} \text{Training set} &:= \text{group}\{1, 2, 3, 4, 5\} \setminus \{i\} , \\ \text{Validation set} &:= \text{group}\{i\} ; \quad i = 1, \dots, 5 . \end{aligned} \tag{3.8}$$

In other words, we train on 80% of the data, and validate on the remaining 20%, 5 different times. The resulting measures of fit are recorded so as to compute the average and error. In general, one can perform n -fold cross-validation, but 5-fold is standard. This way, we avoid both over-fitting, in that the validation set is never seen during training, as well as sample bias, in that we have trained and validated on various different combinations of the data.

Several technical comments are at hand. First, we are going up to graphs with 100 nodes, whose adjacency matrices have 10^4 (albeit sparse) entries. This becomes computationally too intensive for a laptop computer. Thus, for the following, we will take the first 5000 of the database (we still employ the full set for statistics and for some unsupervised ML later). As can be seen from Figure 2 (a1), the vast majority of the available graphs are at small number of vertices (this is purely for the easy of computation and compilation in the database). In fact, up to 5000, the largest is 25 nodes. We subsequently pad all adjacency matrices of the graphs with zeros to the right and to the bottom so that all A_i are 25×25 .

Second, it is important that equivalent representation of the input be built-in. Indeed, given a graph, any relabeling of the vertices is equivalent: this amounts to the *same* row/column permutation of the adjacency matrix ². We typically perform 20 random permutations to each adjacency matrix A and assign then the same property, where increasing the size of the data to 10^5 (in practice, the number is smaller due to possible same random permutations). As a technical aside, we perform the

² This should be contrasted with Cayley multiplication table of finite groups, as done in [HK], where *independent* row and column permutations are allowed.

permutations *before* the zero-padding since this clearly makes more sense.

Finally, we will shuffle the data completely so that the sizes of the graphs are not ordered. This is to avoid the bias of seeing only graphs of a certain number of vertices and validating against those of a different number. Of course, one could purposefully do this so as to try to extrapolate to more complicated graphs from simpler ones, as was done in the spirit of [BHJM]. We leave discussions on this extrapolation to later. For now, in the 5-fold cross validation, we will always be training on an assortment of graphs of varying sizes and complexity.

3.1.1 A Graphical Miscellany

Before we move on to the chief quantity of our concern, viz., the graph Laplacian. Let us warm-up with machine-learning of some well-known properties of graphs (again, q.v. the Appendix for some explicit examples).

Planarity: One of the most important properties of a graph is whether it is **planar**, in other words, whether the graph can be embedded into a plane and therefore can be drawn so that no edges cross except meeting at the nodes. This a clear binary classification problem: adjacency matrix $A \rightarrow$ yes/no for planarity, well adapted to our supervised learning philosophy.

At our 5-fold validation, we find that

$$P_N \simeq 0.812 \pm 0.004, F_1 \simeq 0.832 \pm 0.004, \phi \simeq 0.619 \pm 0.009, \quad (3.9)$$

using a logistic regression classifier (several other methods were tested but this simple regression seems to be the optimal). In other words, having seen 20% of the data, the ML has predicted (in under a minute on an ordinary laptop) whether a graph is planar by “looking” at the adjacency matrix, to rather good confidence.

Now, there is a generalization of planarity, which is called **graph skewness**. This is the minimal number of edges to remove which would render the G planar. We remark that one could try to set up the complete prediction of the skewness. However, the variation here is enormous within our dataset and ranges from 0 (planar) to many

thousands. We could instead make a 3-category classification: 0 (planar), 1 (skewness = 1) and 2 (skewness > 1). Doing so gives a slightly less significant result than the above, with P_N and ϕ dropping down to 0.747 ± 0.005 and 0.597 ± 0.008 (note that F_1 is not defined for non-binary classifications). Upon examining the confusion matrix, we see that the majority of mis-classifications is due to differentiating 1 and 2. In other words, it is a little more difficult for the ML to distinguish precise skewness, and the quick decision on planarity is much easier.

Next, one could analyze the **genus**, i.e., the genus of the Riemann surface onto which G can be embedded. Clearly, genus $g = 0$ means that the graph is planar. Again, we can split the situation into $g = 0$ (planar), $g = 1$ (doubly periodic) or $g > 1$. This is in parallel to the trichotomy of Riemann surfaces, whether as complex varieties, they are Fano, Calabi-Yau, or general type, admitting, respectively, positive, zero, or zero curvature. Here, we permute each adjacency matrix within the 3 categories by 10, 20 and 180 respectively in order to reach a more balanced data-set of around 16K each. It turns out that this division is very much amenable to machine-learning, and we find at our 5-fold validation, that

$$P_N \simeq 0.814 \pm 0.003, \quad \phi \simeq 0.721 \pm 0.005, \quad (3.10)$$

which is even better than planarity recognition (again, there is no F_1 -score to report since it is a ternary classification here).

Chromatic Number: The minimal number of colours needed to colour the vertices such that no two adjacent vertices share the same colour is the chromatic number and is another important combinatorial quantity. Here, like skewness, the chromatic number has a large variation within our dataset. Thus, for starters, we can split this into a binary classification problem: whether the chromatic number ch is, say, > 2 or ≤ 2 . We remark that this distinction of $ch \leq 2$ or not is important, because it is inequivalent to whether G is **bipartite**³.

Now, within \mathcal{D} , those with $ch > 2$ dominates, so we will do more permutation enhancements for $ch \leq 2$ (we do 5 permutations for each $ch > 2$ and 30 for $ch \leq 2$, giving a more balanced set of around 30K for each category). At our 5-fold validation,

³i.e., whether all vertices of the graph can be split into two disjoint sets such that each edge connects two vertices, one from each such set; the study of bipartite graphs have become important in theoretical physics, from brane-tilings in AdS/CFT to SYM amplitudes.

then we find

$$P_N \simeq 0.773 \pm 0.005, F_1 \simeq 0.772 \pm 0.005, \phi \simeq 0.548 \pm 0.009, \quad (3.11)$$

using a random forest classifier, which was found to be optimal. This performance is, interestingly, worse than machine-learning planarity.

In light of the four-colour theorem, it is expedient⁴ to focus on classification chromatic number for planar graphs. This can be turned into a 3-category classification: chromatic number 2, 3, or 4 for a planar graph within our data set. We enhance the data by permuting 30, 10 and 30 times respectively for the 2, 3, 4 chromatic categories, giving around 10K for each. Trying on most standard classifiers, regressors, as well as neural networks with sigmoid activation functions, does not seem to produce results more significant than $\phi \simeq 0.5$.

Diameter: To give a notion of how big a graph is, one typically uses the **diameter** of G : first, find the shortest distance between any pair of vertices (i.e., the minimal number of edges forming a path that is needed to go from one to the other), then, the diameter is the maximum amongst all these distances. Since this has a large variation for our data-set, we can roughly divide the diameter $D(G)$ to be in 3 categories: $D(G) \leq 2$, $3 \leq D(G) \leq 4$, and $D(G) > 4$. Permuting the first two categories by 10 and the third by 30 to give about 30K per category, we have a roughly balanced 3-category classification problem. Then, at our 5-fold validation, we find

$$P_N \simeq 0.765 \pm 0.004, \phi \simeq 0.647 \pm 0.005, \quad (3.12)$$

using a gradient boosted tree classifier, which was found to be optimal. Again, the performance is quite good and the computation, under a minute per epoch of training.

Girth: Another important measure of the size of the graph is the **girth**,

$$\text{Girth}(G) := \text{minimum over the lengths of all cycles in } G; \quad \infty \text{ if } G \text{ is acyclic.} \quad (3.13)$$

⁴In terms of graphs, the statement is that the chromatic number of any planar graph is at most 4 (cf. [AHK]).

Within our dataset, the girth varies from 3 (around 4500 graphs), 4 (around 1400), and more than 4 (around 1400, including the acyclics). Enhancing the data with random permutations for these 3 categories by 5, 30 and 40 respectively give a fairly balance set of around 15-20k for each of the 3 categories. A classifier is then trained (optimized between nearest-neighbour and decision-tree) and our 5-fold validation gives

$$P_N \simeq 0.771 \pm 0.017, \phi \simeq 0.656 \pm 0.026, \quad (3.14)$$

in a matter of minutes.

As a parallel problem, one could consider the binary classification of

$$\text{Girth}(G) = \text{ or } \neq \infty. \quad (3.15)$$

This is a fundamental characteristic of G because it tells whether it is acyclic (possessing any cycles). While there are nice algorithms such as topological sort which decides this in polynomial time, let us see how such a problem responds to ML. We find our 5-fold validation to give an extremely good behaviour of

$$P_N \simeq 0.954 \pm 0.001, F_1 \simeq 0.955 \pm 0.001, \phi \simeq 0.912 \pm 0.002, \quad (3.16)$$

using a gradient-boost decision tree.

Special Cycles: Two classic problems concerning graphs are cycles which traverse all vertices and edges. Indeed, if a cycle traverses all edges exactly once, it is an **Eulerian** cycle⁵. We can see whether machine-learning can distinguish graphs which possess Eulerian cycles or not as a binary classification. In our database of 5000 graphs, randomly permuting the negatives 5 times and the positives 30 times gives a roughly balanced set of around 17K cases each. At our 5-fold validations, we find that a random forest classifier obtains

$$P_N \simeq 0.731 \pm 0.015, F_1 \simeq 0.721 \pm 0.026, \phi \simeq 0.473 \pm 0.024, \quad (3.17)$$

which is again rather good.

Similarly, a cycle which traverses all vertices exactly once is called a **Hamiltonian**

⁵Indeed, Euler's 1736 translation of the Königsberg bridge problem to the study of such cycles started the subject of graph theory.

cycle. Again, we can turn our dataset into a binary classification problem of whether machine-learning can tell which graphs have a Hamiltonian cycle by looking at the adjacency matrix. It should be emphasized that this is known to be an NP-hard problem so stochastically learning a classifier is important. Here, random permutation of the negatives 15 times and the positives 6 times gives a fairly balanced set of around 20K cases each. Using a random forest classifier, we find that

$$P_N \simeq 0.781 \pm 0.008, F_1 \simeq 0.770 \pm 0.009, \phi \simeq 0.564 \pm 0.017, \quad (3.18)$$

which is as encouraging as deciding the presence of Eulerian cycles.

3.1.2 Maximal Laplacian Eigenvalue

Having warmed up with the exploration of a collage of graphical properties to see how well they respond to machine-learning - with many having encouraging results - we now turn to the object of our main concern, viz., the Laplacian. We will investigate L first before proceeding to the normalized Laplacian Δ in the next section.

It is expedient to study the bounds on the Laplacian spectrum. The lower, as shown earlier, is 0, so let us turn to the upper bound. Within our sample of the first 5000 graphs, we first show a histogram, in Figure 3, of the **maximal eigenvalue** of the Laplacian, rounded to the nearest integer. The reason for this round-off will be explained momentarily. A good fit of the distribution is found to be the Landau distribution:

$$p(x) = \int_0^\infty \left(\frac{t}{\sigma}\right)^{-\frac{2t}{\pi}} \sin(2t) e^{\frac{t(\mu-x)}{\sigma}} dt, \quad \mu \simeq 6.204, \quad \sigma \simeq 1.456. \quad (3.19)$$

The reason we have taken the integer round-off is so that we can establish a discrete classification (we will move on to address the full, continuous problem shortly). As can be seen from the distribution, there is a large variation here, consisting of all integers from 2 to 25. In a matter of seconds on an ordinary laptop, using a logistic regression classifier for this 24-category problem, we find that our 5-fold validation gives

$$P_N \simeq 0.4982 \pm 0.007, \phi \simeq 0.415 \pm 0.007. \quad (3.20)$$

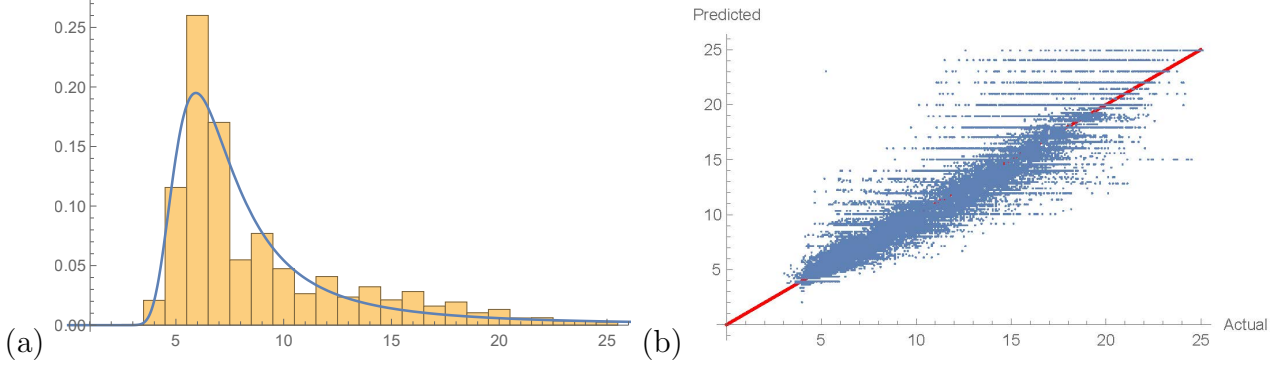


Figure 3: (a) The histogram of the round-off of the maximal Laplacian eigenvalue for the first 5000 of our Wolfram database of undirected simple graphs.; (b) The linear fit of the maximal Laplacian against that predicted by the NN in (3.22) from the adjacency matrix.

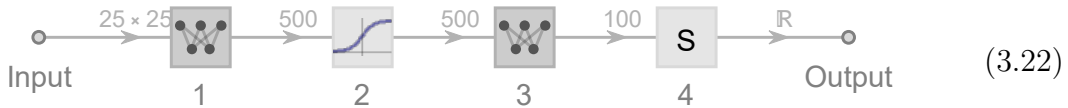
In other words, the regressor has predicted the correct values of the integer round of the maximal Laplacian eigenvalue to about 50% with almost as much confidence.

Moving onto the full problem, we need to establish a predictor for maximal Laplacian spectrum in the form of

$$A_{ij} \longrightarrow \max_{\lambda \text{ Eigenvalue}} \lambda(L) . \quad (3.21)$$

Of course, due to the high dimensionality of the input variables: there are ${}^{25}C_2 = 300$ degrees of freedom for the adjacency matrix A_{ij} , to find a simple function f which, by some non-linear regression, that produces the maximal λ , would be rather difficult.

We establish a neural network (NN) in the form of a forward-propagating, 4 layer perceptron (MLP) which was found to very efficient in computing cohomology [He] and which has the form



That is, the input layer is a 25×25 matrix, followed by a fully-connected linear layer taking it to 500 nodes, each of which is passed to a Sigmoid activation function $x \rightarrow (1 - e^{-x})^{-1}$. These 500 neural nodes are then mapped to 100 nodes in a fully connected layer, before finally summed to a real number, which should correspond to the output of the maximal λ .

Using an ADAM optimizer with batch size 64, and with supervised training of the above NN on the 20% of the data, we can plot the predicted versus the actual maximal λ for the unseen validation 80%. This is shown in Part (b) of Figure 3. The scatter plot should be as close to the line $y = x$ (drawn in red) as possible and we do see indeed the predicted maximal λ are so. A best fit shows that

$$y = -0.0619868 + 1.01097x, \quad R^2 = 0.93; \quad (3.23)$$

the closeness of R^2 , the coefficient of determination, to 1, shows that this is a good fit.

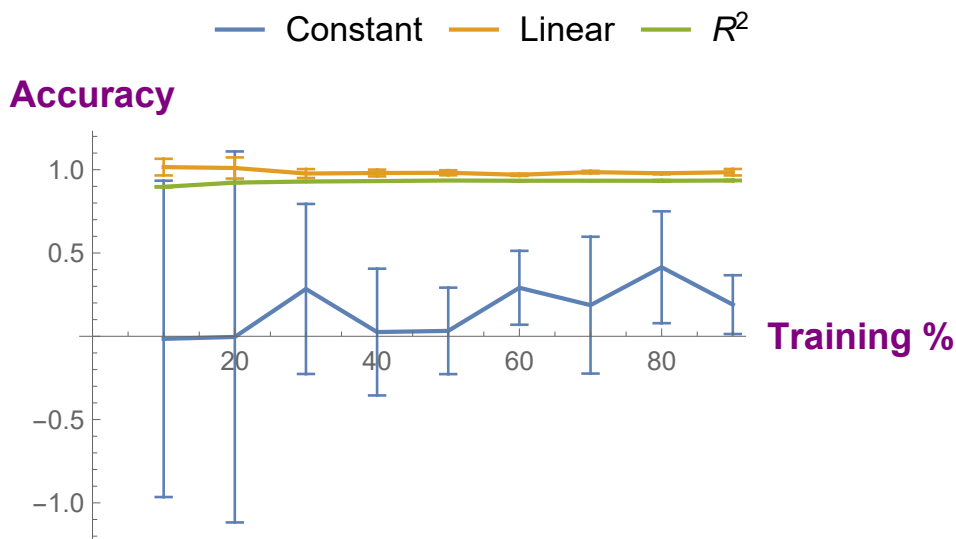


Figure 4: The training curve for the neural network in (3.22) for ML of the maximum Laplacian eigenvalue of our graph dataset. A random sample of $x\%$ is trained and then the NN is validated on the remaining $(100 - x)\%$; we let x range from 10 to 90 in increments of 10. We show 3 measures of goodness of fit: the constant and linear terms of the linear regression of predicted and actual values, as well as the coefficient of determination R^2 .

We can get a glimpse of the machine-learning behaviour also by training on randomly chosen $x\%$ and validating on the complementary $(100 - x)\%$. Plotting the goodness of fit of the ML gives us a **training curve**. This is shown in Figure 4. Training is done from 10% to 90% random sample from the graph database, and validated on the complement. As in the foregoing discussions, there is a list of predicted maximal Laplacian eigenvalues and a list of actual values, we perform linear regression $y = ax + c$ on these to obtain (i) the constant term c , (ii) the linear coefficient a , as well as (iii) the coefficient of determination R^2 . The value of c should be close to 0, a should be close to 1 and R^2 , to 1. We see that starting from a mere 20%, the

behaviour is already very good. The error bars are collected from 5 rounds (epochs) of training: both a and R^2 behave well from the beginning, whilst c fluctuates less and less as we increasing training size.

3.1.3 Spectral Gap

On the other extreme, we can study the lowest Laplacian eigenvalue. Of course, by Eq. (2.4), the lowest eigenvalue is 0. In fact, we have that (cf. [Fan])

PROPOSITION 2. *The dimension of the nullspace of the graph Laplacian is equal to the number of connected components of G .*

Thus, for all our graphs, there is *exactly one* 0-eigenvalue. The next smallest eigenvalue (possibly with multiplicity) is called the (Laplacian) **spectral gap**; it is therefore the first positive eigenvalue of L .

We show the histogram (normalized to a probability) in part (a) of Figure 5, together with its best fit as a probability distribution, which is found to be a generalized Gamma distribution

$$p(x) = (x - \mu)^{\alpha\gamma - 1} \exp(-((x - \mu)/\beta)^\gamma) ; \quad x > \mu ,$$

$$\alpha \simeq 5.164, \quad \beta \simeq 0.031, \quad \gamma \simeq 0.387, \quad \mu \simeq 0.023 . \quad (3.24)$$

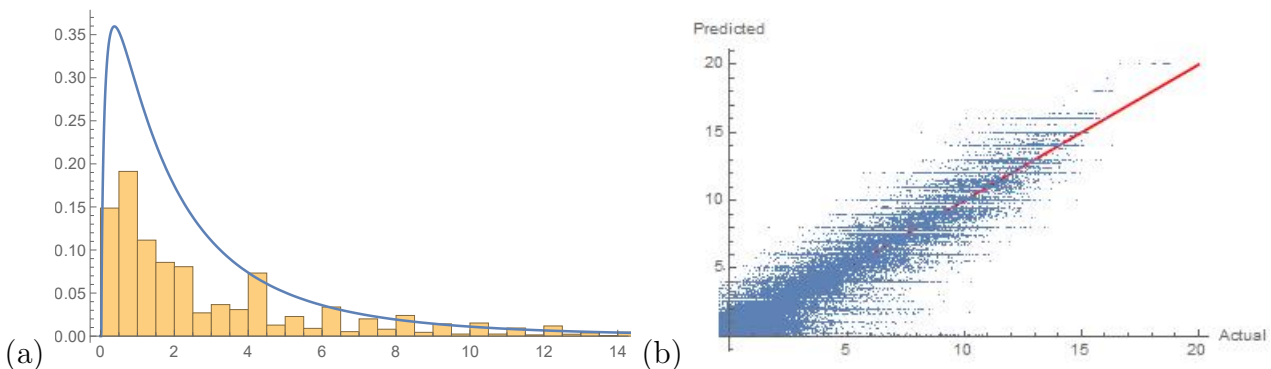


Figure 5: (a) The histogram of the spectral gap (smallest positive Laplacian eigenvalue) for the first 5000 of our Wolfram database of undirected simple graphs; (b) its linear fit against that predicted by the NN in (3.22) from the adjacency matrix.

As with the maximal eigenvalue, we perform ML by the NN in (3.22). At 20% random sample for training and validating against the complement 80%, we can find

the best fit line to be $y \simeq 0.910x + 0.700$ with $R^2 \simeq 0.897$, which is shown in Part (b) of Figure 5. Interestingly, the fit here, whilst still rather good, is less well-behaved than (3.23).

4 Laplacian Spectra and Inequalities

We have now practiced with ML on a plethora of graph theoretic quantities and can continue to study our protagonist the normalized Laplacian Δ . One of the classic results in differential geometry is the Li-Yau theorem on the bound of the eigenvalue of the Laplacian. In particular, we have [LiYau]

THEOREM 2 (Li-Yau). *Let M be a compact Riemannian manifold of dimension n and diameter ⁶ $\delta(M)$, with Ricci curvature bounded below by $(n-1)K$, then the first non-zero eigenvalue of the Laplacian is bounded below by*

$$\lambda \geq \frac{\exp \left[-(1 + \sqrt{1 - 4(n-1)^2 \delta(M)^2 K}) \right]}{2(n-1)\delta(M)^2} .$$

As discussed in §2.2, one of the motivations of studying graph Laplacians is to see whether there are such “discrete” analogues of these spectral bounds. Now, it is a standard result in graph theory that the non-zero Laplacian eigenvalues are bounded by (cf. [Fan] and also q.v. survey in [Moh])

$$\lambda \geq [\delta(G)V(G)]^{-1} , \tag{4.1}$$

where $\delta(G)$ is the diameter of the graph G and $V(G) := \sum_{v_i} d(v_i)$ is the volume of G , the sum over all degrees of all vertices.

This was improved by Lin-Yau [LY, LY2] to be

$$\lambda \geq [d(G)\delta(G) \exp(d(G)\delta(G) + 1) - 1]^{-1} , \tag{4.2}$$

where, as was in Theorem 1, $d(G) := \sup_{x \in V} d_x$ is the supremum over all vertex degrees.

⁶ The diameter of a manifold is the supremum over all geodesic lengths on M .

4.1 Eigenvalue Distributions of the Normalized Laplacian

To get an idea of the Laplacian spectrum, especially the lowest and the highest, for our database, we first order all the graphs from left to right in complexity (roughly the number of vertices and edges, as ordered in the Wolfram database) across the abscissa, then, on each point, we list all the Laplacian eigenvalues starting from the lowest to the highest (we recall from Proposition 2 that the lowest is a single 0 as all our graphs are connected). This is shown in Part (a) of Figure 6. We see that all eigenvalues here are between 0 and 2. The upper bound of 2 is a known result for the normalized Laplacian (equaling to 2 for bipartite graphs).

In Part (b) of the figure, we show a comparison between the magnitudes of (1) the smallest positive eigenvalue, the spectral gap λ_{min} , (2) the RHS of the standard bound (4.1), and (3) the RHS of the Lin-Yau bound (4.2). Due to the presence of the exponential, we show the log of the 3 quantities. We see how λ_{min} dominates, as is required. Now, it might appear that the standard bound (4.1) is stronger than that of (4.2). However, we need to remember that the latter applies to *locally* finite graphs which could have an infinite number of vertices and the subsequent sum could grow dramatically.

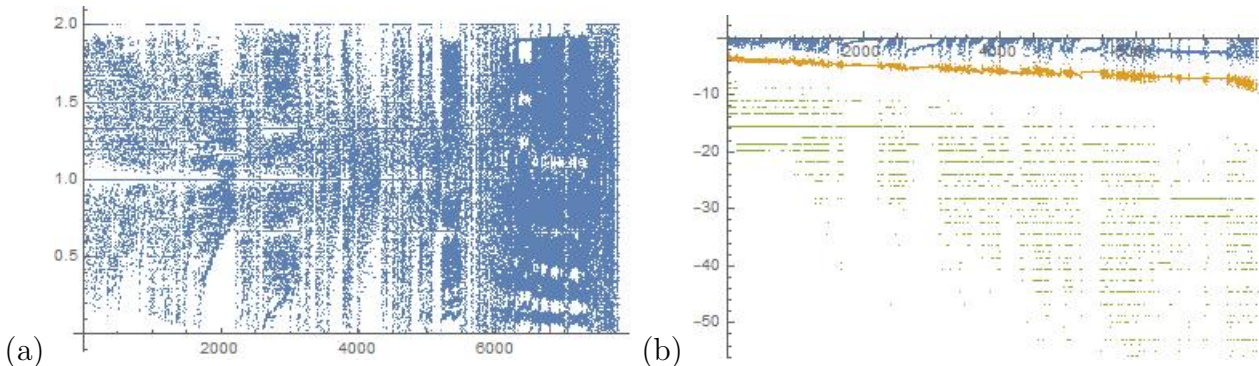


Figure 6: (a) all eigenvalues for the 7785 connected simple graphs from the Wolfram database, sequentially from left to right in the order of the database (roughly according to complexity); (b) the minimal non-zero eigenvalue (spectral gap) of the the 7785 graphs, compared to the two bounds in (4.1) and (4.2).

4.2 Machine-Learning Spectral Bounds

As in §3.1.3 for the non-normalized Laplacian L , we can now apply ML to the normalized Laplacian Δ of our concern, especially in light of the spectral inequalities of

the Li-Yau/Lin-Yau type.

4.2.1 Unsupervised Treatments

We begin by applying some unsupervised techniques in order to better visualize the eigenvalue distribution. The diagram in part (a) of Figure 6 is unenlightening. For one thing, the x-axis is an arbitrary ordering (part (b), of course, is meaningful in showing the comparison between the inequalities). A more natural coordinate to take, for instance, is the ordered Laplacian eigenvalues. To normalize, we left pad all matrices to the maximal dimension without dataset, viz, 100. This way, we have a **point cloud** $\Lambda \subset \mathbb{R}_{\geq 0}^{100}$, each point of which is

$$\vec{\lambda} := (0, 0, \dots, 0, \lambda_1, \lambda_2, \dots, \lambda_k)_{100}, \quad \lambda_{i-1} \leq \lambda_i; \quad \mathbb{R}^{100} \supset \Lambda = \{\vec{\lambda}\}. \quad (4.3)$$

This point cloud is of enormous dimension, but luckily unsupervised ML provides precisely the technique to visualize it: via the method of **principle component analysis** (PCA). We can map \mathbb{R}^{100} to, for example, \mathbb{R}^2 . The dimensionality reduction is shown in part (a) Figure 7; it is an interesting semi-crescent shape. As an illustration, we have separated the planar versus the non-planar graphs. It is curious that the Laplacian spectrum of the non-planar graphs lie a little above that of the planar ones. In part (b) of the same figure, we isolated the planar graphs (of which there are 2933). For these, the chromatic number can be 2, 3, or 4 due to the 4-colour theorem. We perform a similar (100 \rightarrow 2)-dimensional reduction, and mark the 3 different chromatic numbers. Here we still obtain a crescent shape but the 3 categories of chromatic numbers do not seem that much different.

Another way to visualize high dimensional data is to use topological data analysis (TDA) [CZCG] (cf. recent review in [OPTGH]). Here, we compute the persistent homology of the point cloud and present them in terms of so-called **barcodes**. Using the *Julia* implementation of *Eirene* from [Ei], we compute the zeroth and first persistent homology for the point cloud Λ in (4.3); these are shown in parts (a) and (b) respectively in Figure 8. Note that computing barcodes is extremely expensive so around 7000 points in \mathbb{R}^{100} would be quite prohibitive; we subsequently took 500 random sample points to give an idea of the persistence.

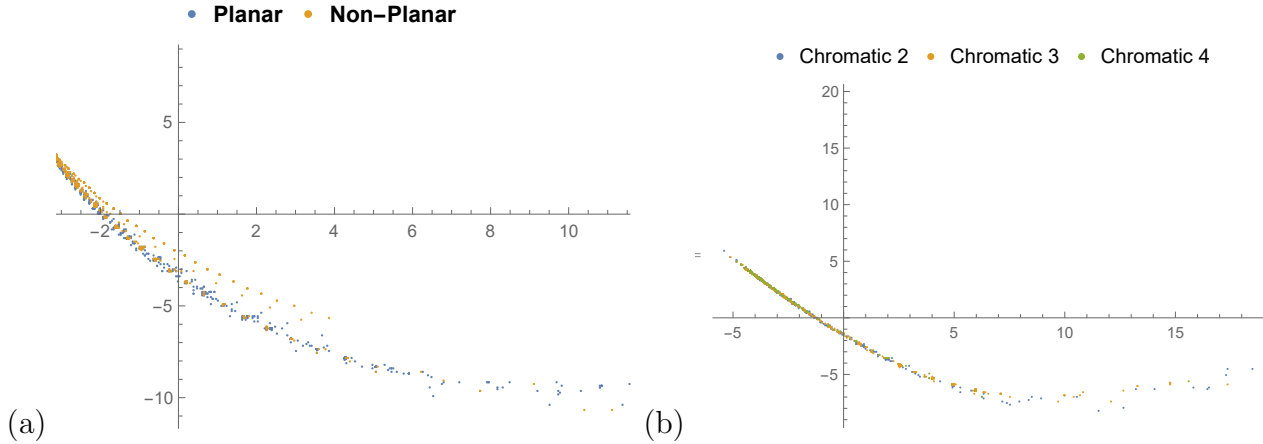


Figure 7: (a) The PCA for our 7785 graphs, reducing the 100-dimensional point-cloud defined by the Laplacian spectrum as in (4.3) to 2-dimension. (b) Similarly, for the 2933 planar graphs, we plot the point-cloud, but distinguished by the 3-categories of chromatic numbers 2, 3, and 4.

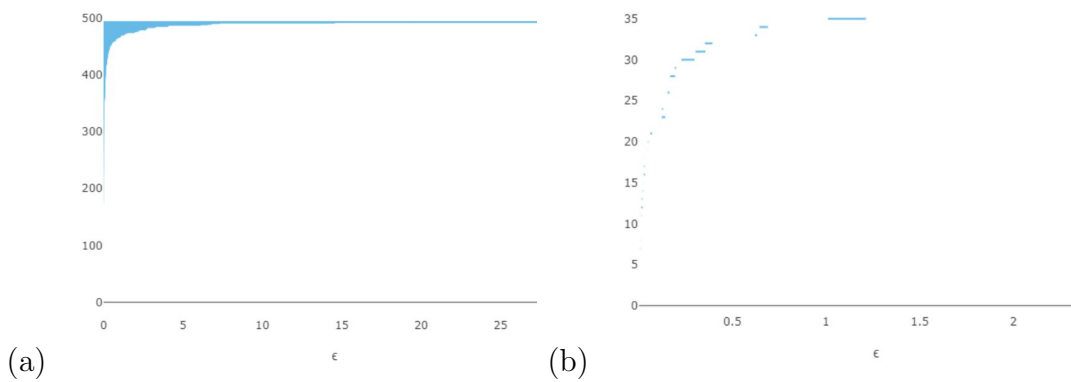


Figure 8: The barcodes for 500 random samples of the 100-dimensional point cloud of the Laplacian spectrum of our graph dataset, indicating the persistent homology in dimension 0 and 1 (respectively parts (a) and (b)).

4.2.2 Supervized Learning

Let us now perform supervised ML on the Laplacian eigenvalue. First, we repeat the analysis in §3.1.3, but now for the normalized Laplacian Δ (rather than the ordinary L which was done there). In other words, we have a labeled problem of

$$A_{ij} \longrightarrow \min_{0 \leq \lambda \text{ Eigenvalue}} \lambda(\Delta), \quad (4.4)$$

from the adjacency matrix to the spectral gap. Suppose the ML has seen only 20 % of the data at random, how does it predict the gap for the remaining 80% ?

We use a slightly improved NN than (3.22) by inserting an element-wise hyperbolic tangent layer between the linear layer and the final output summation layer. Still, the performance is not as good as that of L is §3.1.3. We present the linear fit between the actual and the predicted value on the validation set in part (a) of Figure 9. The line is found to be

$$y \simeq 0.884x + 0.049, \quad R^2 \simeq 0.753. \quad (4.5)$$

In part (b) of the figure, we present the training curve from 10% to 90% training data as before and plot the behaviour of the linear and constant terms of the linear regression fit, as well as the R^2 -coefficient.

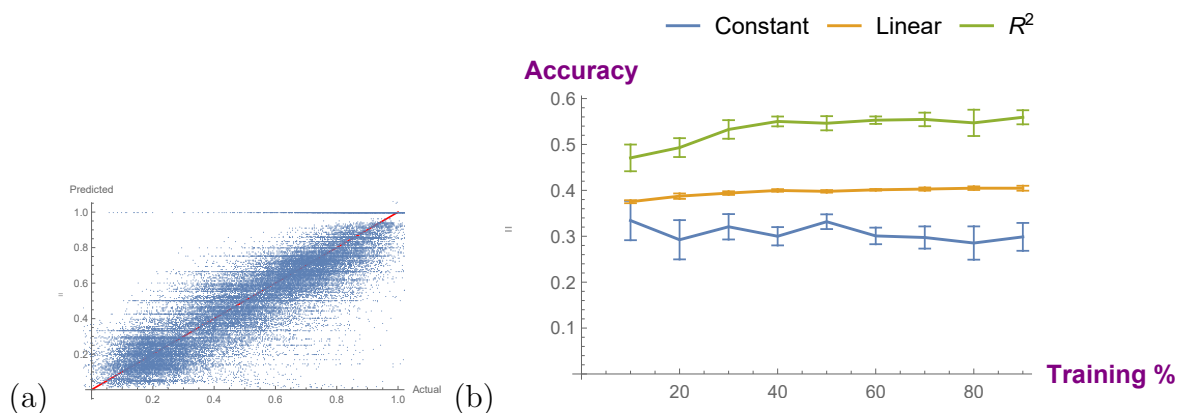


Figure 9: The training curve for the neural network in (3.22) for ML of the maximum Laplacian eigenvalue of our graph dataset. A random sample of $x\%$ is trained and then the NN is validated on the remaining $(100 - x)\%$; we let x range from 10 to 90 in increments of 10. We show 3 measures of goodness of fit: the constant and linear terms of the linear regression of predicted and actual values, as well as the coefficient of determination R^2 .

As a double check against §3.1.2, we also performed the above supervised ML for the maximal eigenvalue of normalized Laplacian Δ . Again, the performance, though satisfactory, is not as impressive as (3.23). Here, at 20-80% split, we have $y \simeq 0.816x + 0.314$ with $R^2 \simeq 0.575$.

It is curious that eigenvalues of the ordinary Laplacian L seem more amenable to ML than our normalized Laplacian Δ ; one reason for this that the former has much larger variation whilst the latter is confined to within $[0, 2]$ and our NN structure is doing non-linear regression without attempting to detect scale.

5 Ricci Flat Graphs

As with Riemannian manifolds, once a notion of curvature is established, one of the first explorations to undertake is Ricci-flatness. For Kähler manifolds, this led to the Calabi Conjecture/Yau Theorem. For (locally finite) graphs, continuing with the idea of Γ in Definition 4, a curvature-dimension type inequality was introduced [LY]:

DEFINITION 5. *A curvature-dimension inequality can be introduced to the graph Laplacian Δ as*

$$\Gamma_2(f, f)(x) \geq \frac{1}{m}(\Delta f(x))^2 + k(x)\Gamma(f, f)(x) ,$$

where $m \in \mathbb{R}_{\geq 1}$ is called the dimension of Δ as an operator and $k(x)$ is the lower bound on the Ricci curvature associated to Δ . Let us call this inequality type $CD(m, k)$.

The next concepts we need are distributions and transport ⁷ on a (locally finite) graph $G = (V, E)$:

DEFINITION 6. *A **probability distribution** over V is a map $\mu : V \rightarrow [0, 1]$ such that $\sum_{x \in V} \mu(x) = 1$. A function $f : V \rightarrow \mathbb{R}$ is called **1-Lipschits** if $f(x) - f(y) \leq d(x, y)$ for each $x, y \in V$, i.e., the difference in value of the function between any two vertices is bounded by the distance between them. The **transportation distance** between two probability distributions μ_1 and μ_2 is*

$$W(\mu_1, \mu_2) := \sup_{f \text{ 1-Lipschits}} \sum_{x \in V} f(x)(\mu_1(x) - \mu_2(x)) .$$

5.1 Two Notions of Ricci-Flatness for Graphs

Thus prepared, we have two notions of Ricci-flatness for graphs and we will proceed with care to compare them ⁸. In some sense, Ricci-flat graphs are the discrete analogues of Calabi-Yau manifolds. It is still an open conjecture by the second author

⁷ The transport distance is usually first defined in terms of a coupling but we will adhere to the one thus presented to minimize introducing new concepts.

⁸There is actually a third notion of Ricci-flatness, which we will not need in this paper. This definition requires the concept of a k -frame as was introduced by [FanYau]; it also requires that the graph be **regular** in that all vertices have the *same degree/valency*. We say that a regular graph $G = (V, E)$ has a **local k -frame** at vertex x if there exist injective mappings $\eta_{i=1, \dots, k}$ from a neighbourhood of x into V such that (1) x is adjacent to $\eta_i x$ for all $1 \leq i \leq k$ and (2) $\eta_i x \neq \eta_j x$ for $i \neq j$. Then, we can define the regular graph G to be Ricci flat at x if there is a local k -frame in the

that in each complex dimension n , there is a finite number (in the sense of topological type) of compact, smooth Calabi-Yau n -folds ⁹

One way to define Ricci-flatness is to modify the notion of Ricci curvature for metric spaces in the sense of [Oll] The classification was addressed in [LLY] (cf. also [CKLLLY, CKLLLY2, OSY]). For vertices $x, y \in V$, and any real value $\alpha \in [0, 1]$, define the probability distribution μ_x^α and using the transportation distance W :

$$\mu_x^\alpha(z) := \begin{cases} \alpha & \text{if } z = x, \\ \frac{1-\alpha}{d(x)}, & \text{if } z \sim x, \\ 0 & \text{otherwise;} \end{cases} \quad k_\alpha(x, y) := 1 - \frac{W(\mu_x^\alpha, \mu_y^\alpha)}{d(x, y)}. \quad (5.1)$$

Then, we have an Ollivier-type Ricci curvature as defined in [LLY]:

DEFINITION 7. *Define curvature $k(x, y) = \lim_{\alpha \rightarrow 1} \frac{k_\alpha(x, y)}{1-\alpha}$. A (locally finite) graph is Ricci-flat if $k(x, y)$ vanishes for any edges $x \sim y$.*

We will refer to this notion as **OLLY-Ricci-flatness**.

Another way to define graph Ricci-flatness is to consider the inequality $CD(m, k)$ in Definition 5 whose classification was addressed in [HuL]:

DEFINITION 8. *A Ricci-flat graph G in the sense of curvature-dimension is one whose Laplacian Δ satisfies $CD(\infty, 0)$.*

We will refer to this definition as **CD-Ricci-flatness**.

neighbourhood of x such that for all $i = 1, \dots, k$ we have a notion of commutativity of transport:

$$\bigcup_{j=1}^k (\eta_j \eta_i)x = \bigcup_{j=1}^k (\eta_i \eta_j)x.$$

One might refer to this third notion as k -frame-Ricci-flatness.

⁹In complex dimension $n = 1$, for instance, there is only T^2 . For $n = 2$, there is only T^3 and the K3 surface. For $n = 3$, the discovered number is huge (on the order of 10^{10}) but the number of topologically inequivalent CY 3-folds is expected to be finite; cf. brief review in [BHHP].

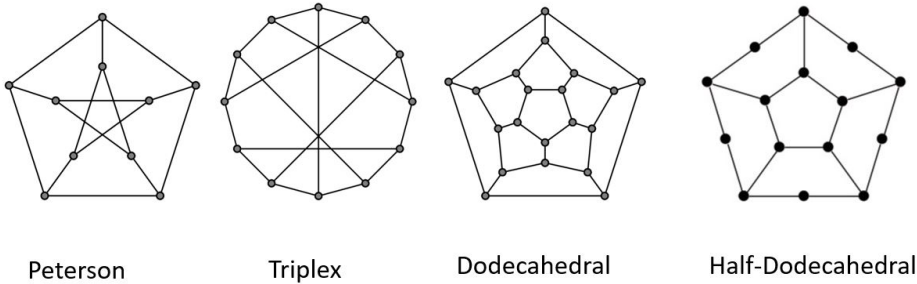
5.1.1 Classification Results

As mentioned, both notions have undergone classification. First, a key result of [LLY] is that whilst for girth (cf. definition in (3.13)) 3 and 4, there are an infinite number of OLLY-Ricci-flat graphs (q.v. [BLY]), for girth 5 or more, we have that ¹⁰

THEOREM 3 (Lin-Lu-Yau). *If G is a (locally finite) OLLY-Ricci-flat graph with girth ≥ 5 , then it is one of the 2 infinity families:*

- *the infinite line;*
- *the cycle graph $C_{n \geq 6}$;*

or one of the 4 exceptional cases:



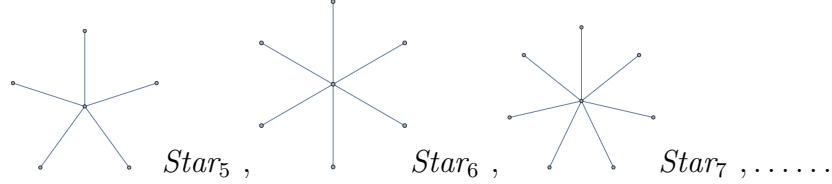
It is further curious that the classification is reminiscent of an ADE pattern: a simple infinite family, a more complicated infinite family, and coincidentally 3 exceptional cases (if we combined the dodecahedral with the half-dodecahedral).

Second, the main result of [HuL] is the classification of CD-Ricci-flat graphs (in cit. *ibid.*, the authors consider weighting and normalization so their results are more general. For our present purposes, the vertex weights m and the edge weights μ are all set to 1 and their theorem 1.3 applies):

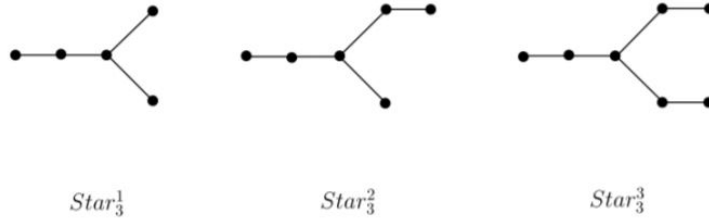
THEOREM 4 (Hua-Lin). *Let $G = (V, E)$ be of girth at least 5, then it satisfies $CD(0, \infty)$ and is thus CD-Ricci-flat if it is one of the following:*

¹⁰Note that the initial classification in [LLY] missed the Triplex graph, which we later found in [CKLLLY2].

- The path graph $P_{k \geq 1}$, the cycle graph $C_{n \geq 5}$;
- The infinite line $P_{\mathbb{Z}}$ or the infinite half-line $P_{\mathbb{N}}$;
- The star graphs $Star_{n \geq 3}$



- The extended star graphs $Star_3^{i=1,2,3}$:



Thus one again, we have a somewhat ADE-type pattern with an infinite family of simple cases (together with the 1/2-infinite versions), an infinite family of less-simple cases (the star graphs), as well as 3 exceptionals (incidentally, the three extended star graphs are precisely the Dynkin diagrams for D_5 , E_6 and \widehat{E}_6).

Question: A question immediately comes to mind as to whether AI can distinguish a Ricci flat graph just by looking at it. A similar venture was undertaken in [He, HL, HK]. In particular, in [HL], the question of whether ML can tell if a Calabi-Yau manifold is elliptically fibred was posed, and answered in the affirmative. There, by going over the data-set of complete intersection Calabi-Yau manifolds in products of projective spaces, wherein the elliptic fibrations have been found using tradition techniques in algebraic geometry [AGGL], a NN was set up and was found to over 99% confidence that such fibration structures can be machine-learned.

Now, non-elliptic manifolds are actually quite rare in the space of Calabi-Yau manifolds and data enhancement in the manner of what we will shortly describe was first performed. The situation is similar here: Ricci-flat graphs in both senses of the

definitions are relatively rare in the space of graphs, especially in the OLLY sense. We will thus enhance the data by performing appropriate permutations of the adjacency matrices, which are clearly equivalent representations.

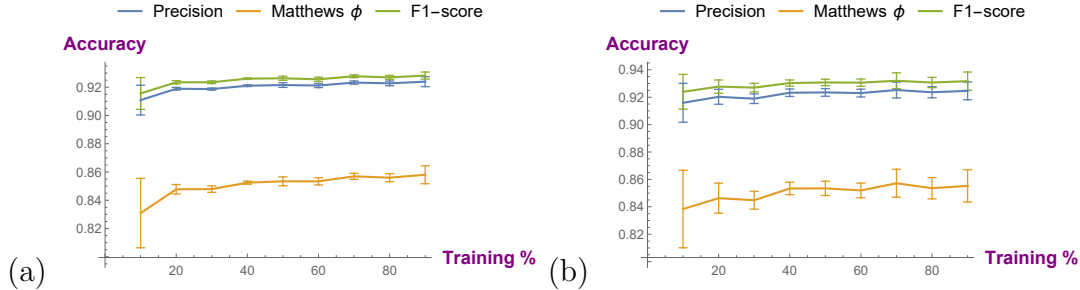


Figure 10: The training curve for a classifier (optimized between a logistic regression and a gradient boosted decision tree) for distinguishing a Ricci-flat graph from our dataset, respectively for the OLLY and CD notions of the Ricci-flatness in parts (a) and (b). A random sample of $x\%$ is trained and then the NN is validated on the remaining $(100 - x)\%$; we let x range from 10 to 90 in increments of 10. We show 3 measures of goodness of fit: the naive precision, the F1-score and the Matthews coefficient.

5.2 Distinguishing OLLY-Ricci-Flat Graphs with ML

In our dataset, we have 1806 graphs of girth at least five ¹¹. Amongst these let us consider the OLLY-Ricci-flat ones: the infinite line is out since we only consider finite graphs, so we have the cycles graphs and the 4 exceptional cases, totaling 20.

We enhance the data by permuting the adjacency matrices of the Ricci-flat cases by 800 times randomly, and that of the non-Ricci-flat cases by 8 times randomly. This gives us a rather balanced labeled set of around 15000 each of “1” (Ricci flat) and “0” (not Ricci flat), perfectly adapted for a binary classification.

At 5-fold cross-validation and using a optimized classifier of gradient-boosted decision trees and logistic regression, we find that

$$P_N \simeq 0.925 \pm 0.004, F_1 \simeq 0.927 \pm 0.004, \phi \simeq 0.855 \pm 0.007, \quad (5.2)$$

which is excellent. The training curve is shown in part (a) of Figure 10. We see that distinguishing OLLY-Ricci-flat graphs is performed very well by ML, in that we are

¹¹There are 1805 built in, including the cycle graphs, the dodecahedron, the Triplex, and the Petersen, but the half-dodecahedron is added in by hand.

into the 90’s in terms of measures of goodness of fit. As an extra precaution, what if we *randomly* assigned 1 to around 50% of the data and 0 to the remainder? The 5-fold cross validation gave $P_N \simeq 0.546 \pm 0.029$, $F_1 \simeq 0.101 \pm 0.205$ and $\phi \simeq 0.004 \pm 0.0135$. This means that the precision is at around 50%, as good as random guessing, which is further confirmed by the F_1 -score and Matthews ϕ -coefficient being near 0. This is re-assuring indeed and shows that there is a significant *pattern* which distinguishes Ricci-flatness, and that the pattern is not random.

5.3 Distinguishing CD-Ricci-Flat Graphs with ML

Having gained confidence, let us move onto the other notion of Ricci-flatness. Within our dataset of 1805 graphs with girth at least 5, there are 58 which satisfy the criterion of CD-Ricci-flatness in accordance with Theorem 4. We thus perform 300 random permutations of the adjacency matrices for the “yes” cases and 5 for the “no” cases, giving us about 9000 each, of a roughly balanced 18000 database ready for binary classification.

At 5-fold cross-validation and using a optimized classifier of gradient-boosted decision trees and logistic regression, we find that

$$P_N \simeq 0.924 \pm 0.004, F_1 \simeq 0.931 \pm 0.004, \phi \simeq 0.854 \pm 0.006, \quad (5.3)$$

which is again excellent. The training curve is shown in part (b) of Figure 10.

We conclude that distinguishing Ricci-flat graphs, in both the OLLY and the curvature-dimension notions, is performed comparably well by ML, in that we are into the 90’s in terms of measures of goodness of fit by an optimized binary classifier. The high percentage agreement is further checked by the high F1-score and Matthews coefficient, showing that the classification is truly good and that false-positives and false-negatives are insignificant.

6 Homology and Cohomology of Graphs

A fundamental result in Riemannian geometry is the decomposition of Hodge which implies that (co-)homology of the manifold should be described by the zero-modes of the Laplacian. Can this be carried over to our graphical context? In [GLMY1, GLMY2], a notion of differential forms and boundary operators, and hence cohomology and homology, were introduced on **directed** simple graphs.

In brief, the construction proceeds as follows.

- Let $G = (V, E)$ be a finite directed graph (digraph) so that $V = \{i\}_{i=1, \dots, n}$ are the vertices and $E = \{i_k i_{k+1}\}$ are directed edges (arrows) from vertex i_k to i_{k+1} . On G , we have
 - **Elementary p -path** $e_{i_0 \dots i_p}$, which is any ordered sequence of $p+1$ vertices i_0, i_1, \dots to i_p ;
 - Elementary **regular** p -path is one for which $i_k \neq i_{k+1}$, so that there is no back-tracking;
 - **Allowed** elementary p -path is one for which $i_k \rightarrow i_{k+1} \in E$ for all $k = 0, \dots, p-1$, i.e., the path is actually traversed by arrows in G ;
- Fix \mathbb{K} to be a commutative ring with unity (we mostly just take \mathbb{R}) and consider the free \mathbb{K} -module generated by the elementary p -paths:

$$\Lambda_p := \left\{ \sum k e_{i_0 \dots i_p} \right\} = \text{Span}_{\mathbb{K}} \{ \text{elementary } p\text{-paths} \}; \quad (6.1)$$

elements of Λ_p are just called p -paths.

- Define the boundary operator $\partial : \Lambda_{p+1} \rightarrow \Lambda_p$ by

$$\partial e_{i_0 \dots i_p} := \begin{cases} \sum_{q=0}^p (-1)^q e_{i_0 \dots \hat{i}_q \dots i_p} & \text{for } p \geq 1, \\ 0 & \text{for } p = 0, \end{cases} \quad (6.2)$$

where the hat means omission and we set $\Lambda_{-1} = \{0\}$;

- Then we have $\partial^2 = 0$;

– Define the subspace of regular paths

$$\mathcal{R}_p := \text{Span}_{\mathbb{K}}\{\text{regular elementary } p\text{-paths}\} \subset \Lambda_p , \quad (6.3)$$

which is linearly isomorphic to the quotient of Λ_p by the irregular paths; i.e., we set all irregular paths to 0.

– Define one further step the space of allowed paths

$$\mathcal{A}_p := \text{Span}_{\mathbb{K}}\{\text{allowed regular elementary } p\text{-paths}\} \subset \mathcal{R}_p \subset \Lambda_p \quad (6.4)$$

and consider the ∂ -invariant subspaces (i.e., the boundaries are still allowed paths)

$$\Omega_p := \{v \in \mathcal{A}_p : \partial v \in \mathcal{A}_{p-1}\} \subset \mathcal{A}_p . \quad (6.5)$$

By construction, the 0 and 1-paths are just the vertices and arrows respectively:

$$\Omega_0 = \mathcal{A}_0 = V, \quad \Omega_1 = \mathcal{A}_1 = E , \quad (6.6)$$

so $\dim \Omega_0 = |V|$ and $\dim \Omega_1 = |E|$. Then, we have the chain complex

$$0 \leftarrow \Omega_0 \xleftarrow{\partial} \Omega_1 \xleftarrow{\partial} \dots \xleftarrow{\partial} \Omega_p \xleftarrow{\partial} \dots \quad (6.7)$$

from which can define **graph homology groups** $H_p(G) := \ker(\partial_p) / \text{Im}(\partial_{p+1})$.

- Next, we can define the dual to the above. First, a p -form on G is a \mathbb{K} -valued function on V^{p+1} , i.e., it is a function $\omega(i_0, \dots, i_p)$ with $p + 1$ arguments. We have the freely generated \mathbb{K} -module

$$\Lambda^p := \text{Span}_{\mathbb{K}}\{\mathbb{K}\text{-valued function } \omega\} = \left\{ \sum_{i_0, \dots, i_p \in V} \omega(i_0, \dots, i_p) e^{i_0 \dots i_p} \right\} , \quad (6.8)$$

where $e^{i_0 \dots i_p}$ is a canonical basis of Λ^p .

– On Λ^p we can define an exterior derivative $d : \Lambda^p \rightarrow \Lambda^{p+1}$ as

$$(d\omega)(i_0, \dots, i_{p+1}) := \sum_{q=0}^{p+1} (-1)^q \omega(i_0, \dots, \hat{i}_q, \dots, i_p) , \quad (6.9)$$

where the hat is again omission of the index. We can check that $d^2 = 0$.

– Again, regularity can be defined by having no back-tracking of indices:

$i_k \neq i_{k+1}$, so that we have the subspace of regular p -forms

$$\mathcal{R}^p := \text{Span}_{\mathbb{K}}\{e^{i_0 \dots i_p} : i_k \neq i_{k+1}\} \subset \Lambda^p . \quad (6.10)$$

- We now wish to quotient out by the non-allowed p -forms, corresponding to the non-allowed p -path:

$$\mathcal{N}^p := \text{Span}_{\mathbb{K}}\{e^{i_0 \dots i_p} : i_0 \dots i_p \in \Lambda_p \setminus \mathcal{A}_p\} . \quad (6.11)$$

- Subsequently we can define $\Omega^p := \mathcal{R}^p / (\mathcal{N}^p + d\mathcal{N}^{p-1})$ by treating non-allowed p -forms as zero and check that Ω^p is d -invariant. This gives us the dual complex

$$0 \rightarrow \Omega^0 \xrightarrow{d} \Omega^1 \xrightarrow{d} \dots \xrightarrow{d} \Omega^p \xrightarrow{d} \dots \quad (6.12)$$

from which we can define **graph cohomology groups** $H^p(G) = \ker(d_p) / \text{Im}(d_{p-1})$.

As usual with duality between homology and cohomology (of compact smooth manifolds), a non-degenerate bilinear pairing can be established between $H_p(G)$ and $H^p(G)$, rendering them isomorphic as dual vector spaces. In particular, they have the same dimension. Note that

$$\begin{aligned} \dim H_p(G) &= \dim \Omega_p - \dim \partial\Omega_p - \dim \partial\Omega_{p+1} = \\ \dim H^p(G) &= \dim \Omega^p - \dim d\Omega^p - \dim d\Omega^{p-1} . \end{aligned} \quad (6.13)$$

Whence a graph Euler number can be defined as

$$\chi(G) = \sum_{p=0}^n (-1)^p \dim H_p(G) , \quad (6.14)$$

with n sufficiently large so that $\dim H_{p>n}(G) = 0$. Due to the alternating sum in (6.13), as always, we have an Euler-Poincaré type of relation

$$\chi(G) = \sum_{p=0}^n (-1)^p \dim \Omega_p(G) . \quad (6.15)$$

6.1 Machine Learning Graph Euler Number

We leave a detailed calculation of graph homology to the Appendix, demonstrating its non-triviality. For manifolds, the application of machine-learning to topological invariants was initiated in [He]. It was found, for example, that Hodge numbers of classes of Calabi-Yau manifolds respond well to a simple neural network. Naturally, one could ask whether a similar behaviour occurs here.

First, we need to establish a reasonable dataset. We can take our database of undirected graphs, and assign random directions since the above notion of homology requires orientation. To be precise, we take a selection of 10 undirected graphs (up to 6 vertices since the homology computation gets quite intensive) in the set, and assign 100 random directions to each, and randomly permute the $\chi = 0$ (which are slightly under-represented) adjacency matrices 48 times and the $\chi \neq 0$ cases 20 times. This gives us a binary classification problem of around 15K in each category. Interestingly, neither a classifier (optimized between logistic regression and a decision tree) nor a neural network of the type in (3.22) could do this problem well: we obtain naive precision around 0.50 with $\phi \simeq 0.08$, which is marginally better than random guessing. We tried other NNs such gated recurrence networks, which performed slightly better at $\phi \simeq 0.15$ but could not find any ML which did so nicely as distinguishing properties such as Ricci-flatness, planarity, chromaticism, etc, as above.

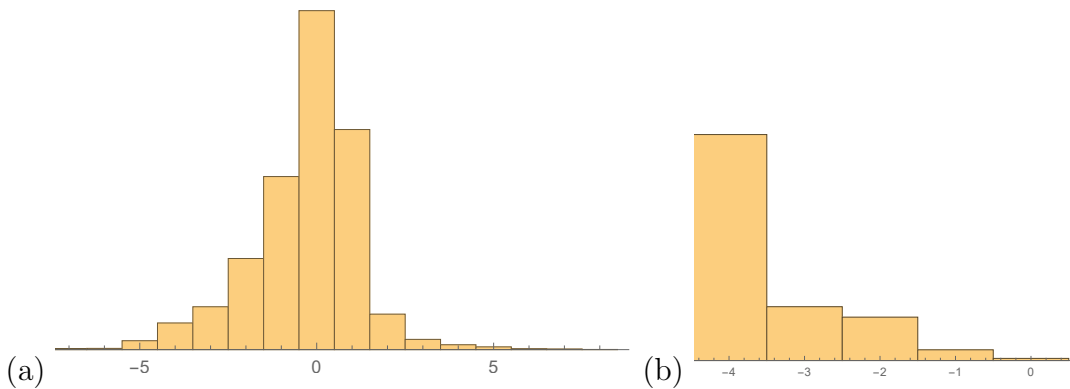


Figure 11: (1) histogram for χ of 10K random graphs with random orientations; (2) histogram for χ of 10K random orientations on the cube.

It is curious that learning such topological quantities for graph did not behave as brilliant as the counterpart in manifolds [He, BHJM]. For reference, we give an idea of the distribution of graph Euler number in Figure 11. In part (a), we take the first 100 graphs in the dataset, with 100 random direction assignments to each, giving us

10K graphs for which we compute the Euler number and show the distribution, a somewhat Gaussian around 0. In part (b), we take a particular graph with 8 vertices, say the cube, and then assign 5K random orientations to compute the Euler numbers, and show the distribution.

7 Conclusions and Prospectus

Due to their combinatorial nature and hence the facility of representation as matrix manipulations, finite graphs present a natural venue for a diverse number of fields in mathematics and physics. Inspired by a long programme of the second author to investigate discrete analogues of the foundational problems in Riemannian geometry via finite graphs, as well as a more recent programme of the first author to explore how different mathematical structures can be machine-learned or detected by generic neural networks without a priori knowledge, we have employed some of the latest techniques in data science to study aspects of graph theory in connection to Riemannian geometry.

Taking the freely available Wolfram database of finite simple graphs up to 100 nodes as a concrete playground - a representative set of some 8000 graphs - we have examined a host of relevant statistical and machine-learning properties. We warmed up with machine-learning a miscellany of basic graph quantities in a supervised learning paradigm where we trained a classifier/neural network with the labeling “adjacency matrix \rightarrow property”. These are mostly done with random-forest decision trees and logistic regressions that have no knowledge of the underlying graph theory. Denoting the triple-check for accuracy measures as (naive precision, F1-score, Matthews ϕ coefficient) in the case of binary classification or a double-check as (naive precision, Matthews ϕ coefficient) in the case of multi-category classification, we summarize the 5-fold cross-validation results in Table 1. For clarity of comparison we order from best performance downward.

Next, we attempted a more refined predictor to study the maximal and minimal eigenvalues of the Laplacian. Using a simple neural network with the architecture given in (3.22), with standard activation functions, we attempted to predict these eigenvalue bounds by looking solely at the adjacency matrix. The performance is again very good, in that the actual and predicted values fit to a line $y = 1.01x - 0.06$

Property	Accuracy Measure
whether graph is acyclic	(0.954 \pm 0.001 , 0.955 \pm 0.001 , 0.912 \pm 0.002)
whether graph is OLLY-Ricci flat	(0.925 \pm 0.004 , 0.927 \pm 0.004, , 0.855 \pm 0.007)
whether graph is CD-Ricci-flat	(0.924 \pm 0.004 , 0.931 \pm 0.004 , 0.854 \pm 0.006)
whether graph is genus ≤ 0 , $= 0$ or > 0	(0.814 \pm 0.003 , 0.721 \pm 0.005)
whether graph is planar	(0.812 \pm 0.004, 0.832 \pm 0.004, 0.619 \pm 0.009)
girth of graph $= 2$, $= 4$ or > 4	(0.771 \pm 0.017 , 0.656 \pm 0.026)
diameter ≤ 2 , $= 3, 4$ or > 4	(0.765 \pm 0.004 , 0.647 \pm 0.005)
skewness of graph $= 0$, $= 1$ or > 1	(0.747 \pm 0.005 and 0.597 \pm 0.008)
whether there is a Hamilton cycle	(0.781 \pm 0.008 , 0.770 \pm 0.009 , 0.564 \pm 0.017)
whether there is a Eulerian cycle	(0.731 \pm 0.015 , 0.721 \pm 0.026 , 0.473 \pm 0.024)
Round of Laplacian eigenvalue	(0.4982 \pm 0.007 , 0.415 \pm 0.007)
whether graph χ is 0	(0.502 \pm 0.022 , 0.603 \pm 0.054 , 0.078 \pm 0.015)

Table 1: Summary of the accuracy measures (goodness of fitness) for the various quantities as machine-learnt by a decision-tree classifier or simple feed-forward neural network within a 5-fold cross-validation. The triple applies to cases of binary classification and refers to (naive precision, F1-score, Matthews ϕ coefficient) and the pair applies to multi-category classification and refers to (naive precision, Matthews ϕ coefficient).

with $R^2 = 0.93$ (perfect prediction would mean $y = x$ with $R^2 = 1$) for the maximal eigenvalue. Interestingly, the minimal positive eigenvalue (the spectral gap) behaved a bit worse in the prediction, with $y = 0.91x + 0.70$ with $R^2 = 0.897$.

Thus prepared, we studied more precise bounds on the (random-walk normalized) Laplacian [LY, LY2] which are analogues of the classical Li-Yau [LiYau] inequalities for Riemannian manifolds. We studied the statistics of the eigenvalue distributions in comparison with the various inequalities and then applied principal component analysis (PCA) and topological data analysis (TDA) on this distribution. We found that planar and non-planar graphs have rather distinct principal components in the Euclidean point cloud of ordered Laplacian eigenvalues. Then, using supervised learning on the adjacency matrices, a fairly satisfactory prediction of the spectral gap using the neural network was achieved.

Continuing in this vain, bearing in mind the intimate relation between curvature, Laplacian zero-modes and (co-)homology for manifolds, we proceeded to ask the question whether ML could distinguish Ricci-flat graphs. There are two no-

tions of Ricci-flatness for (locally) finite graphs, in terms of OLLY and of curvature-dimension inequalities. We found that supervised ML can achieve more than 90% accuracy (see Table 1 for the precise measures of goodness of fit) for both notions in a 5-fold cross-validation. This is very assuring indeed. It should be emphasized that all ML calculations involved in this paper are performed on a standard laptop using Mathematica, in a matter of seconds or minutes. Finally, we explored the homology of graphs in the sense of [GLMY1, GLMY2]. We presented a rough distribution of Euler number over our database and attempted to machine-learn whether a graph is Euler number 0, though here the accuracies are not high and the ML did not perform much better than random guessing.

We hope our preliminary investigations have paved the road for countless new ventures in applying machine-learning and data scientific techniques to graphs and manifolds. For example, we have not addressed the myriad of key functions in graph theory such as chromatic polynomials, Ihara zeta functions, etc. In addition to the connections between graphs and manifolds, the study of digraphs in the guise of quivers is a hotly pursued topic in mathematics and physics ranging from cluster algebras to supersymmetric quantum field theories, to string theory (cf. [HeZeta] for graph zeta function and gauge theory as well as machine-learning quiver gauge theories and cluster mutation [BFHHM]). Ultimately, we would like to see where in the hierarchy of complexity do combinatorial theorems on graphs reside as far as AI/ML is concerned and indeed whether new conjectures can be formulated [HeTalk]. The terra incognita where ML meets pure mathematics beckons her ever-alluring invitation.

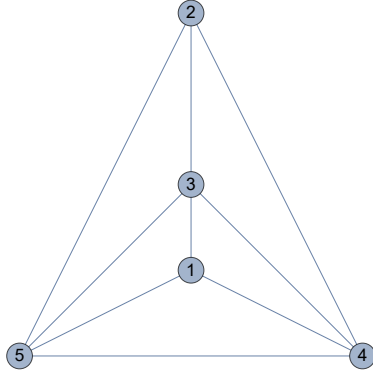
Acknowledgments

We are grateful to Shuliang Bai and Yong Lin for their careful reading of and valuable comments on preliminary drafts of the paper. Y.-H. H is indebted to the Science and Technology Facilities Council, UK, for grant ST/J00037X/1 as well as a chair professorship from Nankai University where this work began. The work of S.-T. Y. is supported in part by a grant from the Simons Foundation in Homological Mirror Symmetry.

A Illustrative Example

In this appendix, let us take a specific example from our database, and compute all the relevant quantities discussed throughout the main body. The particulars thus detailed should serve an illustrative purpose.

Take the so-called **dipyramid graph** $G_{dp}(5)$ on 5 vertices, whose figure, adjacency and degree matrices are shown as follows:



$$A = \begin{pmatrix} 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 \end{pmatrix}, \quad D = \text{Diag}(3, 3, 4, 4, 4),$$

(A.16)

where the vertices have been labeled explicitly. The Laplacian L and the random-walk normalized Laplacian Δ are subsequently

$$L = D - A = \begin{pmatrix} 3 & 0 & -1 & -1 & -1 \\ 0 & 3 & -1 & -1 & -1 \\ -1 & -1 & 4 & -1 & -1 \\ -1 & -1 & -1 & 4 & -1 \\ -1 & -1 & -1 & -1 & 4 \end{pmatrix}, \quad \Delta = D^{-1}L = \begin{pmatrix} 1 & 0 & -\frac{1}{3} & -\frac{1}{3} & -\frac{1}{3} \\ 0 & 1 & -\frac{1}{3} & -\frac{1}{3} & -\frac{1}{3} \\ -\frac{1}{4} & -\frac{1}{4} & 1 & -\frac{1}{4} & -\frac{1}{4} \\ -\frac{1}{4} & -\frac{1}{4} & -\frac{1}{4} & 1 & -\frac{1}{4} \\ -\frac{1}{4} & -\frac{1}{4} & -\frac{1}{4} & -\frac{1}{4} & 1 \end{pmatrix}.$$

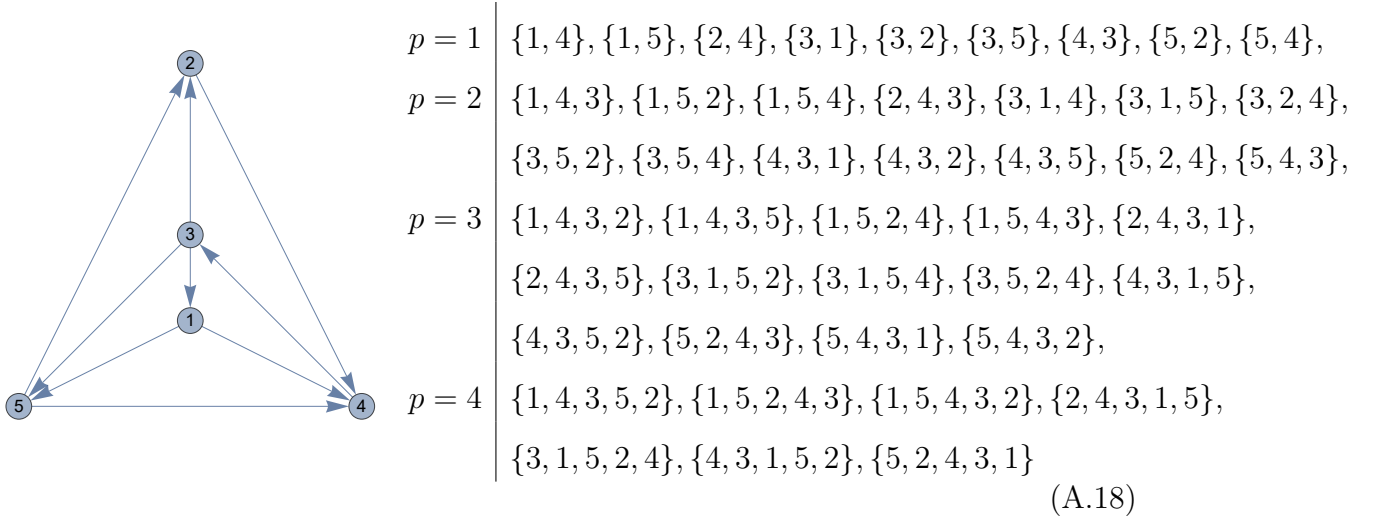
(A.17)

The eigenvalues of Δ are, therefore, $\{\frac{3}{2}, \frac{5}{4}, \frac{5}{4}, 1, 0\}$.

Clearly, the skewness of $G_{dp}(5)$ is 0 and the graph is planar; so too is the genus 0. The chromatic number is 4 and it requires 4 colours, say with vertices 1, 3, 4, 5 with 4 different colours and vertex 2 the same as 1. The diameter is 2, exemplified by the minimal distance from vertex 1 to 2. The girth is 3, since the minimal cycle is length 3. There are no Euler cycles but there are 6 Hamiltonian cycles: $\{1, 3, 2, 4, 5\}, \{1, 3, 2, 5, 4\}, \{1, 3, 4, 2, 5\}, \{1, 3, 5, 2, 4\}, \{1, 4, 2, 3, 5\}, \{1, 4, 3, 2, 5\}$.

We now assign a random direction to $G_{dp}(5)$ as follows, and for reference, we

include all the elementary paths of each length:



Now, whilst [GLMY1, GLMY2] give certain theorems on computing the (co-)homologies for classes of graphs, let us approach the problem by brute force for simplicity.

Let us consider \mathcal{A}_2 , whose basis is given (we introduce formal variables x for convenience) as

$$\mathcal{A}_2 = \text{Span} \left\{ x_{\{1,4,3\}}, x_{\{1,5,2\}}, x_{\{1,5,4\}}, x_{\{2,4,3\}}, x_{\{3,1,4\}}, x_{\{3,1,5\}}, x_{\{3,2,4\}}, x_{\{3,5,2\}}, x_{\{3,5,4\}}, x_{\{4,3,1\}}, x_{\{4,3,2\}}, x_{\{4,3,5\}}, x_{\{5,2,4\}}, x_{\{5,4,3\}} \right\} \quad (\text{A.19})$$

from which we can compute the action of ∂ on each of the regular elementary 2-paths as:

$$\begin{aligned} \partial \mathcal{A}_2 = \text{Span} \{ & -x_{\{1,3\}} + x_{\{1,4\}} + x_{\{4,3\}}, -x_{\{1,2\}} + x_{\{1,5\}} + x_{\{5,2\}}, -x_{\{1,4\}} + x_{\{1,5\}} + x_{\{5,4\}}, \\ & -x_{\{2,3\}} + x_{\{2,4\}} + x_{\{4,3\}}, x_{\{1,4\}} + x_{\{3,1\}} - x_{\{3,4\}}, x_{\{1,5\}} + x_{\{3,1\}} - x_{\{3,5\}}, \\ & x_{\{2,4\}} + x_{\{3,2\}} - x_{\{3,4\}}, -x_{\{3,2\}} + x_{\{3,5\}} + x_{\{5,2\}}, -x_{\{3,4\}} + x_{\{3,5\}} + x_{\{5,4\}}, \\ & x_{\{3,1\}} - x_{\{4,1\}} + x_{\{4,3\}}, x_{\{3,2\}} - x_{\{4,2\}} + x_{\{4,3\}}, x_{\{3,5\}} + x_{\{4,3\}} - x_{\{4,5\}}, \\ & x_{\{2,4\}} + x_{\{5,2\}} - x_{\{5,4\}}, x_{\{4,3\}} - x_{\{5,3\}} + x_{\{5,4\}} \} . \end{aligned}$$

These need to be compared to the basis for \mathcal{A}_1 , which are

$$\mathcal{A}_1 = \text{Span} \left\{ x_{\{1,4\}}, x_{\{1,5\}}, x_{\{2,4\}}, x_{\{3,1\}}, x_{\{3,2\}}, x_{\{3,5\}}, x_{\{4,3\}}, x_{\{5,2\}}, x_{\{5,4\}} \right\} . \quad (\text{A.20})$$

We see that many variables appear in $\partial \mathcal{A}_2$ which are *not* in \mathcal{A}_1 , which means that

as it stands, the basis for $\partial\mathcal{A}_2$ is not ∂ -invariant. In particular, these “bad” variables are

$$x_{bad} = \{x_{\{1,2\}}, x_{\{1,3\}}, x_{\{2,3\}}, x_{\{3,4\}}, x_{\{4,1\}}, x_{\{4,2\}}, x_{\{4,5\}}, x_{\{5,3\}}\} . \quad (\text{A.21})$$

The question then is: what linear combinations of $\partial\mathcal{A}_2$, if any, exists so that when expanded out, all coefficients of the bad coefficients vanish? The dimension of such linear combination of solutions is then the dimension of Ω_2 , the requisite ∂ -invariant subspace of \mathcal{A}_2 . There are 14 terms in $\partial\mathcal{A}_2$ so we can set 14 arbitrary coefficients, imposing that when expanded the coefficients of the 8 x_{bad} vanish gives a linear system which solves explicitly to give us 6 free coefficients, which means that $\dim\Omega_2 = 6$. We have encountered the generic situation by coincidence, in general, we have to solve the linear system case by case.

We now repeat the above for $\mathcal{A}_3, \mathcal{A}_4$, etc. In summary, we find that, in addition to the two by construction, viz., $\dim\Omega_0 = \dim\mathcal{A}_0 = |V| = 5$ and $\dim\Omega_1 = \dim\mathcal{A}_1 = |E| = 9$,

$$\dim\Omega_2 = 6, \quad \dim\Omega_3 = 2, \quad \dim\Omega_{n \geq 4} = 0, \quad (\text{A.22})$$

so that the Euler number is

$$\chi(G_{dp}(5)) = 5 - 9 + 6 - 2 = 0 . \quad (\text{A.23})$$

References

- [ABH] L. Alessandretti, A. Baronchelli, Y. H. He, “ML meets Number Theory: The Data Science of Birch-Swinnerton-Dyer,” arXiv:1911.02008 [math.NT].
- [ACHN] R. Altman, J. Carifio, J. Halverson and B. D. Nelson, “Estimating Calabi-Yau Hypersurface and Triangulation Counts with Equation Learners,” JHEP **1903**, 186 (2019) doi:10.1007/JHEP03(2019)186 [arXiv:1811.06490 [hep-th]].
- [AGGL] L. B. Anderson, X. Gao, J. Gray and S. J. Lee, “Fibrations in CICY Threefolds,” JHEP **1710**, 077 (2017) [arXiv:1708.07907 [hep-th]].
- [AHK] Appel, Kenneth; Haken, Wolfgang; J. Joch, “Every Planar Map is Four-Colorable”, Contemporary Maths, 98, AMS (1989).
- [AHO] A. Ashmore, Y. H. He and B. Ovrut, “Machine learning Calabi-Yau metrics,” arXiv:1910.08605 [hep-th].
- [BCDL] C. R. Brodie, A. Constantin, R. Deen and A. Lukas, “Machine Learning Line Bundle Cohomology,” arXiv:1906.08730 [hep-th].
- [BFHHM] J. Bao, S. Franco, Y. H. He, E. Hirst, G. Musiker and Y. Xiao, “Quiver Mutations, Seiberg Duality and Machine Learning,” [arXiv:2006.10783 [hep-th]].
- [BHHP] J. Bao, Y. H. He, E. Hirst and S. Pietromonaco, “Lectures on the Calabi-Yau Landscape,” arXiv:2001.01212 [hep-th].
- [BHJM] K. Bull, Y. H. He, V. Jejjala and C. Mishra, “Machine Learning CICY Threefolds,” Phys. Lett. B **785**, 65 (2018) [arXiv:1806.03121 [hep-th]].
–, “Getting CICY High,” Phys. Lett. B **795**, 700 (2019) 1903.03113 [hep-th].
- [BHMRT] Bernal EA, Hauenstein JD, Mehta D, Regan MH, Tang T. “Machine learning the real discriminant locus”, arXiv:2006.14078.
- [BK] P. Betzler and S. Krippendorf, “Connecting Dualities and Machine Learning,” Fortsch. Phys. **68**, no.5, 2000022 (2020) [arXiv:2002.05169 [physics.comp-ph]].
- [BLY] Bai, Shuliang, Lu, Linyuan, Yau, Shing-Tung, “Ricci-flat graphs with maximum degree at most 4”, 2018.
- [Ca] Calabi, Eugenio, “The space of Kähler metrics”, Proc. Internat. Congress Math. Amsterdam, 2, pp. 206 - 207 (1954)
– “On Kähler manifolds with vanishing canonical class”, in Fox, Spencer, Tucker, *Algebraic geometry and topology. A symposium in honor of S. Lefschetz*, Princeton Mathematical Series, 12, PUP, pp. 78 - 89 (1957).
- [CHKN] J. Carifio, J. Halverson, D. Krioukov and B. D. Nelson, “Machine Learning in the String Landscape,” JHEP **1709**, 157 (2017) doi:10.1007/JHEP09(2017)157 [arXiv:1707.00655 [hep-th]].
- [CHLZ] H. Y. Chen, Y. H. He, S. Lal and M. Z. Zaz, “Machine Learning Etudes in

- Conformal Field Theories,” [arXiv:2006.16114 [hep-th]].
- [CLN] B. Chow, P. Lu, Peng, L. Ni, Lei, *Hamilton’s Ricci flow*, GTM, 77 (2006).
- [Coh] Taco Cohen, “Learning transformation groups and their invariants,” PhD thesis, U. Amsterdam, 2013.
- [CKLLLY] David Cushing, Riikka Kangaslampi, Yong Lin, Shiping Liu, Linyuan Lu, Shing-Tung Yau, “Ricci-flat cubic graphs with girth five”, arXiv:1802.02982 [math.CO]
- [CKLLLY2] David Cushing, Riikka Kangaslampi, Yong Lin, Shiping Liu, Linyuan Lu, Shing-Tung Yau, “Erratum for Ricci-flat cubic graphs with girth at least five”, arXiv:1802.02979 [math.CO]
- [CSBXCSH] M. Cranmer, A. Sanchez-Gonzalez, P. Battaglia, R. Xu, K. Cranmer, D. Spergel and S. Ho, “Discovering Symbolic Models from Deep Learning with Inductive Biases,” [arXiv:2006.11287 [cs.LG]].
- [CSS] A. Cole, A. Schachner and G. Shiu, “Searching the Landscape of Flux Vacua with Genetic Algorithms,” JHEP **1911**, 045 (2019) [arXiv:1907.10072 [hep-th]].
- [CZCG] G. Carlsson, A. Zomorodian, A. Collins, and L. Guibas, “Persistence barcodes for shapes,” Intl. J. Shape Modeling, 11 (2005), 149 - 187.
- [DHLL] R. Deen, Y. H. He, S. J. Lee and A. Lukas, “Machine Learning String Standard Models,” [arXiv:2003.13339 [hep-th]].
- [DJ] Xue Ding, Tiefeng Jiang, “Spectral distributions of adjacency and Laplacian matrices of random graphs”, Annals of App. Prob. 2010, Vol. 20, No. 6, pp2086 - 2117, arXiv:1011.2608 [math.PR]
- [DLMS] M. Demirtas, C. Long, L. McAllister and M. Stillman, “The Kreuzer-Skarke Axiverse,” arXiv:1808.01282 [hep-th].
- [DMT] K.Ch. Das, S.A. Mojallal, V. Trevisan, “Distribution of Laplacian eigenvalues of graphs”, Linear Algebra and its Applications 508, 2016, pp48 - 61
- [Ei] Greg Helsenman, Eirene, <http://gregoryhenselman.org/eirene/>
- [Fan] Chung, Fan R. K., *Spectral graph theory* AMS (1997) ISBN 978-0-8218-0315-8.
- [FanYau] Fan Chung and S.-T. Yau, “Logarithmic Harnack inequalities”, Math. Res. Lett. (1996), 793 - 812.
- [GH] J. F. Grimminger and A. Hanany, “Hasse Diagrams for $3d \mathcal{N} = 4$ Quiver Gauge Theories – Inversion and the full Moduli Space,” [arXiv:2004.01675 [hep-th]].
- [GLLY] Chao Gong, Yong Lin, Shuang Liu, Shing-Tung Yau, “Li-Yau inequality for unbounded Laplacian on graphs”, arXiv:1801.06021 [math.DG]
- [GLMY1] A. Grigor’yan, Y. Lin, Y. Muranov, S-T. Yau, “Homologies of path complexes and digraphs”, arXiv:1207.2834

- [GLMY2] A. Grigor'yan, Y. Lin, Y. Muranov, S-T. Yau, "Cohomology of Digraphs and (undirected) Graphs", *Asian J. Math.* Vol 19, No 5, 887-932, 2015.
- [Graph] Wolfram Graph Database: <https://reference.wolfram.com/language/note/GraphDataSourceInformation.html>
- [GM] Robert Grone, Russell Merris, "The Laplacian spectrum of a graph II, *SIAM J. Disc.*", *Math.* 7 (1994), 221 - 229.
- [GRV] T. W. Grimm, F. Ruehle and D. van de Heisteeg, "Classifying Calabi-Yau threefolds using infinite distance limits," arXiv:1910.02963 [hep-th].
- [He] Y. H. He, "Deep-Learning the Landscape," arXiv:1706.02714 [hep-th];
 -. "Machine-learning the string landscape," *Phys. Lett. B* **774**, 564 (2017).
- [HeBook] Y. H. He, "The Calabi-Yau Landscape: from Geometry, to Physics, to Machine-Learning," arXiv:1812.02893 [hep-th]. To appear, Springer.
- [HeTalk] Y. H. He, *Machine-Learning Mathematical Structures*, in Oxford ML meets Physics series, Nov, 2019 <https://www.youtube.com/watch?v=nMP2f14gYzc>
- [HeZeta] Y. H. He, "Graph Zeta Function and Gauge Theories," *JHEP* **1103**, 064 (2011) [arXiv:1102.1304 [math-ph]].
- [HHP] Y. H. He, E. Hirst, T. Peterken, "Machine-Learning Dessins d'Enfants: Explorations via Modular and Seiberg-Witten Curves," [arXiv:2004.05218 [hep-th]].
- [HJN] Y. H. He, V. Jejjala and B. D. Nelson, "hep-th," [arXiv:1807.00735 [cs.CL]].
- [HJP] Y. H. He, V. Jejjala and L. Pontiggia, "Patterns in CalabiYau Distributions," *Commun. Math. Phys.* **354**, no. 2, 477 (2017) [arXiv:1512.01579 [hep-th]].
- [HK] Y. H. He and M. Kim, "Learning Algebraic Structures: Preliminary Investigations," arXiv:1905.02263 [cs.LG].
- [HL] Y. H. He and S. J. Lee, "Distinguishing Elliptic Fibrations with AI," arXiv:1904.08530 [hep-th].
- [HMT] K. Hashimoto, S. Sugishita, A. Tanaka and A. Tomiya, "Deep learning and the AdS/CFT correspondence," *Phys. Rev. D* **98**, no.4, 046019 (2018) [arXiv:1802.08313 [hep-th]].
- [HNR] J. Halverson, B. Nelson and F. Ruehle, "Branes with Brains: Exploring String Vacua with Deep Reinforcement Learning," *JHEP* **06**, 003 (2019) [arXiv:1903.11616 [hep-th]].
- [HuL] Bobo Hua, Yong Lin, "Graphs with large girth and nonnegative curvature dimension condition", arXiv:1608.07000.
- [IMWDR] Iten, Raban, et al. "Discovering physical concepts with neural networks." *Physical Review Letters* 124.1 (2020): 010508.
- [JKP] V. Jejjala, A. Kar and O. Parrikar, "Deep Learning the Hyperbolic Volume of

- a Knot,” arXiv:1902.05547 [hep-th].
- [KS] D. Krefl and R. K. Seong, “Machine Learning of Calabi-Yau Volumes,” Phys. Rev. D **96**, no. 6, 066014 (2017) [arXiv:1706.03346 [hep-th]].
- [KSch] D. Klaewer and L. Schlechter, “Machine Learning Line Bundle Cohomologies of Hypersurfaces in Toric Varieties,” Phys. Lett. B **789**, 438 (2019) [arXiv:1809.02547 [hep-th]].
- [KSy] S. Krippendorf and M. Syvaeri, “Detecting Symmetries with Neural Networks,” [arXiv:2003.13679 [physics.comp-ph]].
- [LC] Lample, Guillaume, and Francois Charton. “Deep learning for symbolic mathematics.” arXiv:1912.01412 (2019).
- [LiYau] P. Li, S.-T. Yau, “Estimates of eigenvalues of a compact Riemannian manifold,” AMS Symp. on the Geometry of the Laplace Operator, U. Hawaii at Manoa, 1979, 205-239.
- [LLY] Yong Lin, Linyuan Lu, S.-T. Yau, “Ricci-flat graphs with girth at least five”, arXiv:1301.0102 [math.CO]
-, “Ricci curvature of graphs”, Tohoku Math. J. (2), 63, 4 (2011), 605 - 627.
- [LS] M. Larfors and R. Schneider, “Explore and Exploit with Heterotic Line Bundle Models,” Fortsch. Phys. **68**, no.5, 2000034 (2020) [arXiv:2003.04817 [hep-th]].
- [LY] Yong Lin, Shing-Tung Yau, “A brief review on geometry and spectrum of graphs”, arXiv:1204.3168 [math.CO]
- [LY2] Yong Lin, Shing-Tung Yau, “Ricci curvature and eigenvalue estimate on locally finite graphs,” Math. Res. Lett/ 17 (2010), 345-358.
- [Matt] Gorodkin, Jan. “Comparing two K-category assignments by a K-category correlation coefficient.” Comp. biology and chemistry 28.5-6 (2004): 367-374.
- [Moh] B. Mohar, “The Laplacian spectrum of graphs”, in *Graph Theory, Combinatorics, and Applications*, Vol. 2, Ed. Y. Alavi, G. Chartrand, O. R. Oellermann, A. J. Schwenk, Wiley, 1991, pp. 871 - 898; https://www.fmf.uni-lj.si/~mohar/Reprints/1991/BM91_GTCA2_Mohar_LaplacianSpectrum.pdf
- [MPV] A. Mttter, E. Parr and P. K. S. Vaudrevange, “Deep learning in the heterotic orbifold landscape,” Nucl. Phys. B **940**, 113 (2019) [arXiv:1811.05993 [hep-th]].
- [Nik] Karalias Nikolaos, “Spectral Graph Theory and Deep Learning on Graphs”, PhD Thesis, Aristotle U. Thessaloniki, <http://ikee.lib.auth.gr/record/294952/files/GRI-2017-20536.pdf>
- [OEISg] Finite simple graphs, <http://oeis.org/A001349>
- [Oll] Y. Ollivier, “Ricci curvature of Markov chains on metric spaces”, J. Funct. Anal. 256 (3) (2009), 810 - 864.

- [OPTGH] Nina Otter, Mason A Porter, Ulrike Tillmann, Peter Grindrod and Heather A Harrington, “A roadmap for the computation of persistent homology”, EPJ Data Science 2017 6:17, ArXiv:1506.08903
- [OSY] Ryunosuke Ozawa, Yohei Sakurai, Taiki Yamada “Geometric and spectral properties of directed graphs under a lower Ricci curvature bound”, arXiv:1909.07715 [math.DG]
- [OT] H. Otsuka, K. Takemoto, “Deep learning & k-means clustering in het. string vacua with line bundles,” JHEP **05**, 047 (2020) [arXiv:2003.11880 [hep-th]].
- [Rue] F. Ruehle, “Evolving neural networks with genetic algorithms to study the String Landscape,” JHEP **1708**, 038 (2017) [arXiv:1706.07024 [hep-th]].
- [She] D. J., Sheskin, Handbook of parametric and nonparametric statistical procedures. Chapman and Hall/CRC (2003).
- [Tsh] Tshitoyan, Vahe, et al. “Unsupervised word embeddings capture latent knowledge from materials science literature.” Nature 571.7763 (2019): 95-98.
- [Wolf] Wolfram Research, Inc. Mathematica 12.0, Champaign, IL (2019).
- [WZ] Y. N. Wang and Z. Zhang, “Learning non-Higgsable gauge groups in 4D F-theory,” JHEP **08**, 009 (2018) [arXiv:1804.07296 [hep-th]].
- [Yau] S.-T. Yau, “Calabi’s conjecture and some new results in algebraic geometry,” Proc. Nat. Acad., USA, 74 (5), pp 1798-9, (1977)
- , “On the Ricci curvature of a compact Kähler manifold and the complex Monge-Ampère equation I”, Comm. Pure and Applied Maths, 31 (3), pp 339-411, (1978).