String Data and Machine Learning

Andre Lukas

University of Oxford

Physics meets ML, July 2020

in collaboration with: Lara Anderson, Callum Brodie, Andrei Constantin, Rehan Deen, James Gray, Yang-Hui He, Seung-Joo Lee, Eran Palti

based on: 1407.4787, 1808.09992, 1906.08730, 1906.08769, 2003.13339

Outline

- Introduction and motivation
- String theory basics
- Can ML help reveal mathematical structures?

  Example: Learning line bundle cohomology
- Can ML help with the large amount of string data?

  Example: Learning string theory standard models
- First steps into RL
- Conclusion

Introduction and Motivation

Why string theory and ML?

- String theory leads to very large data sets

  (latest estimate: 10^{272000} solutions to string theory)

  (W. Taylor, Y. Wang, arXiv:1510.04978)

- Data is ``mathematical", often given in terms of integer-valued tensors

- Different from usual data such as pictures, videos

- Machine learning provides a set of "large-data" techniques

So two basic questions:

- Can ML help reveal mathematical structures within string theory? (Can it be more than a ``black box''?)

  Example: Learning line bundle cohomology

  (C. Brodie, A. Constantin, R. Deen, AL, arXiv:1906.08769)

- Can ML help sort through the large amount of string data?

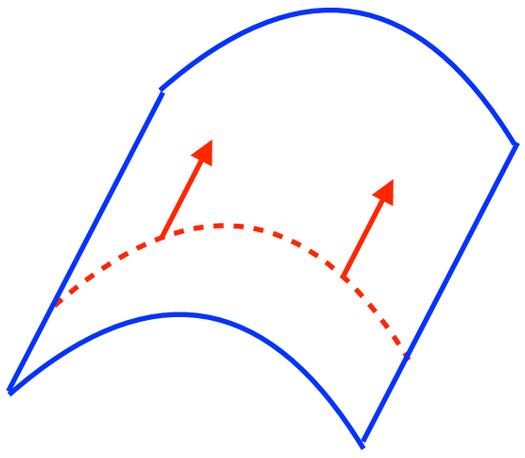
  Example: Learning string theory standard models

  (R. Deen, Y.-H. He, S.-J. Lee, AL, arXiv:2003.13339)

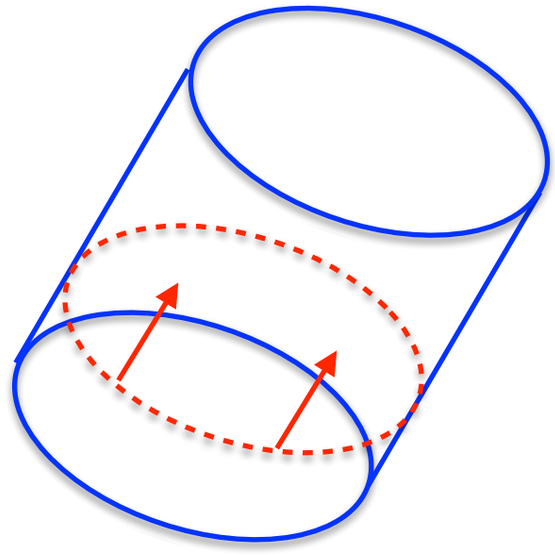
String Theory Basics

Closen and open strings

- open string



closed string



- One free constant: string tension $T = \dfrac{1}{2\pi\alpha'}$

- Consistent in 10 (or 11) space-time dimensions

Spectrum

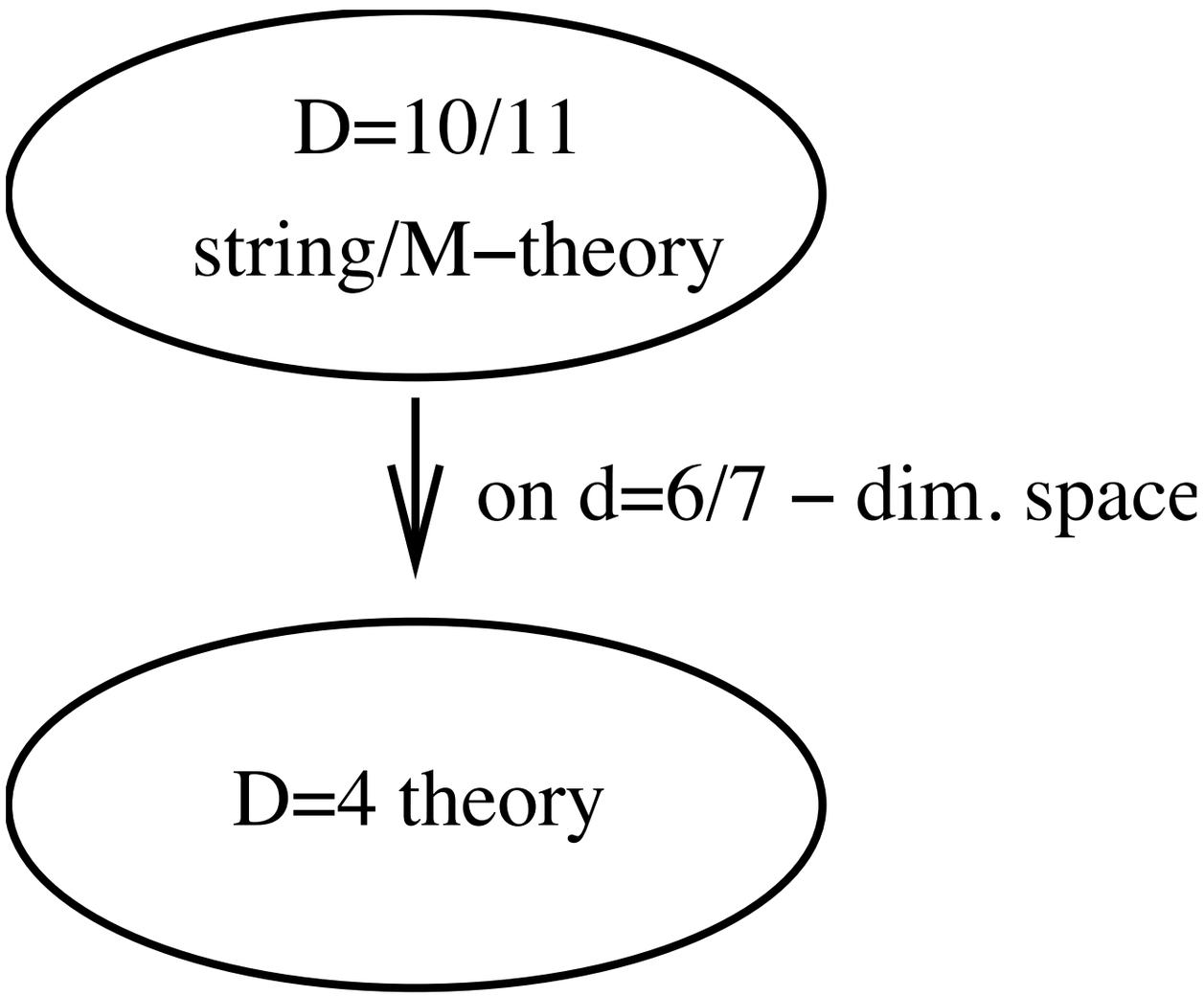
- spectrum: $\alpha' m^2 = n \in \mathbb{N}$ $\begin{cases} n = 0 & \rightarrow & \text{observed particles} \\ n > 0 & \rightarrow & \text{superheavy} \end{cases}$

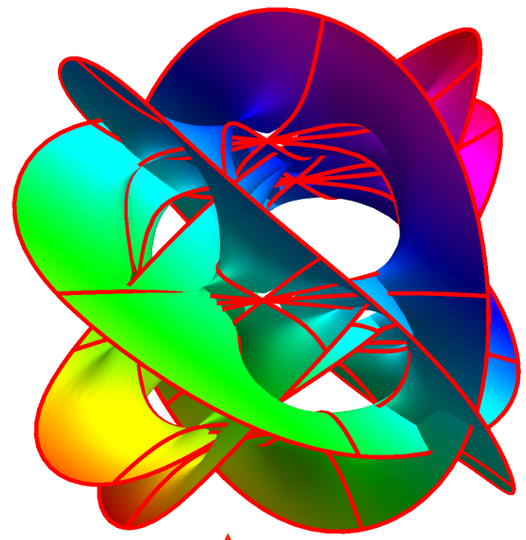
- massless $(n = 0)$ modes contain: graviton (closed string)
  gauge fields (open string)

$$M_s = \frac{1}{\sqrt{\alpha'}} \sim M_{Pl} \sim 10^{19}\text{GeV} > M_U \sim 10^{16}\text{GeV}$$

Dimensions

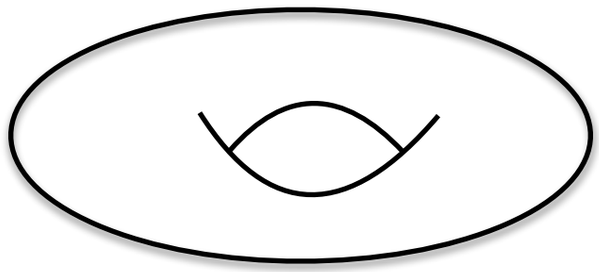
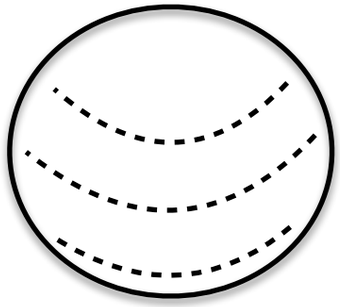
We need to ``curl up" six (or seven) of the dimensions to make contact with physics -> compactification





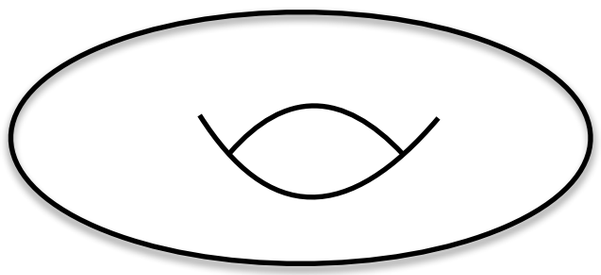
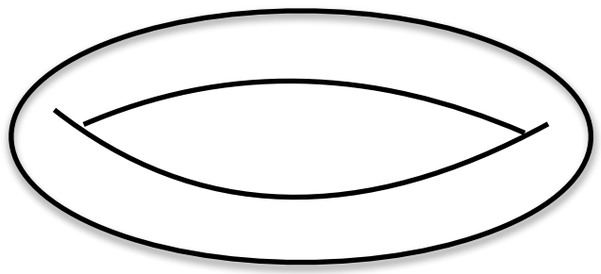
Calabi-Yau manifold
(bi-cubic)

How does the 4d theory depend on the "curling-up"?

topology :  or  ?

-> determines structure of 4d theory: forces, matter content, . . .
(Maths: Algebraic Geometry)

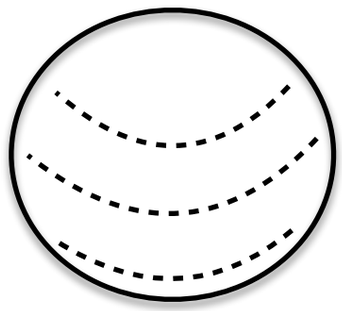
focus for this talk

shape :  or  ?

-> determines couplings/particle masses in 4d theory
(Maths: Differential Geometry)

Topologies for curling up, e.g in 2d:

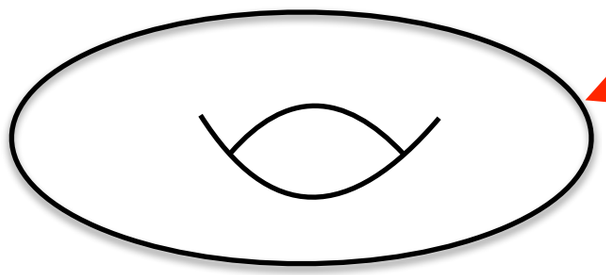
sphere:



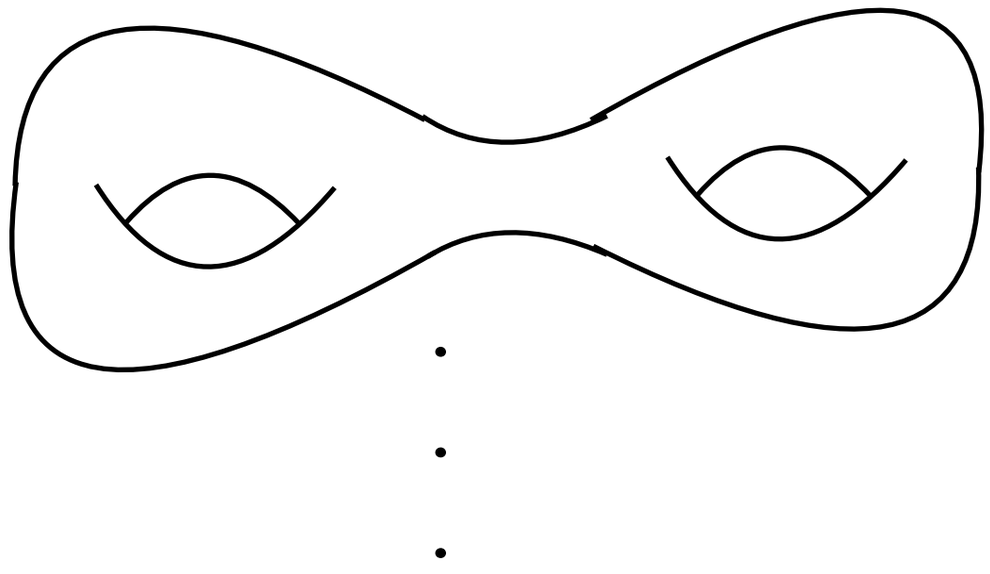
$$g = 0$$

only consistent choice for 2d curling-up

torus:



$$g = 1$$



$$g = 2$$

In 2d topology is classified by the genus g = ``number of holes".

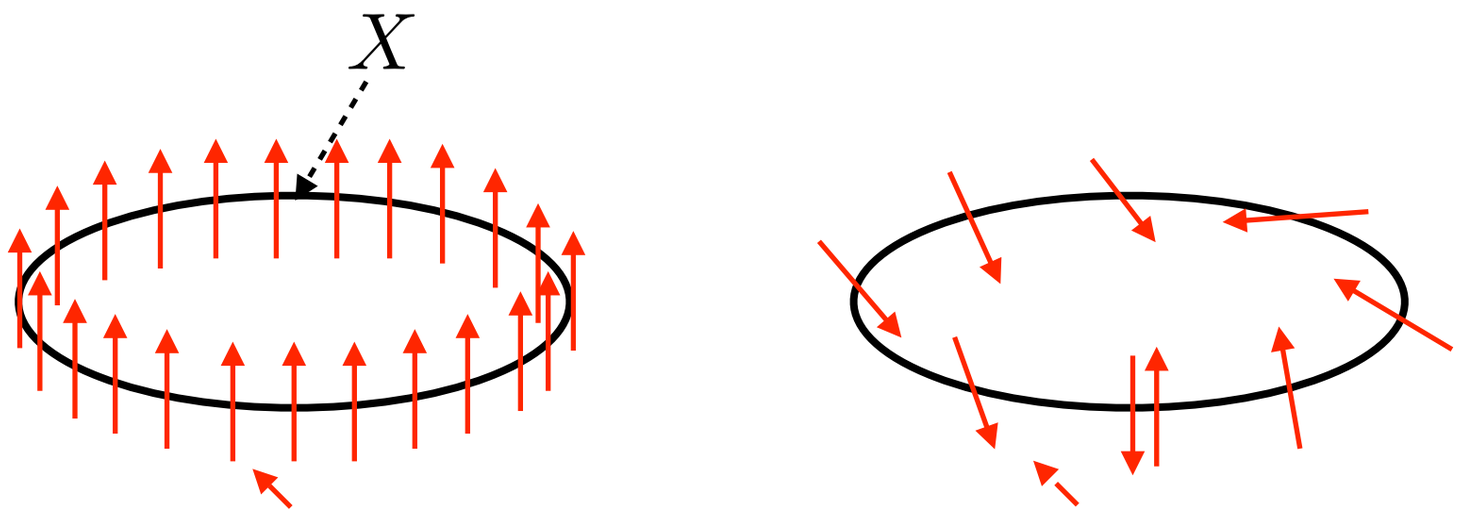
More generally, in 6d, it is classified by integer data -> many choices.

The large number of string solutions mentioned earlier counts these different topologies!

Some choices lead to a 4d theory close to the standard model of particle physics, many others do not.

How to find the 4d theory from a given topology?

The space X carries additional structure, for example line bundles.

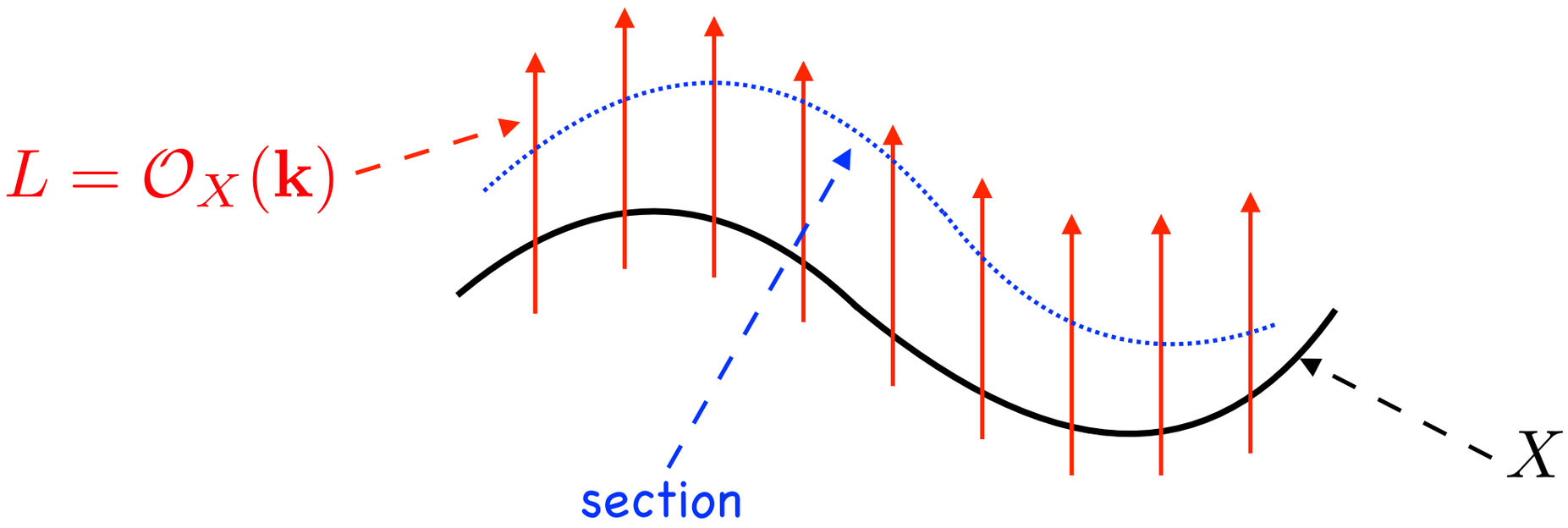


$\mathcal{O}_X$ $\mathcal{O}_X(1)$ $\mathcal{O}_X(2)$ $\cdots$

Line bundle topology is specified by integers

$$\mathcal{O}_X(\mathbf{k}) = \mathcal{O}_X(k_1, k_2, \ldots, k_n)$$

Line bundles have sections:



Number of independent section is counted by cohomology.

A horrendous calculation

$$L = \mathcal{O}_X(\mathbf{k}) \longrightarrow (h^0(L), h^1(L), h^2(L), h^3(L))$$

Counts #particles in 4d

Can ML help reveal mathematical structures?

Computing line bundle cohomology

Base manifold: complex surface or (CY) three-fold X

Line bundles: $L = \mathcal{O}_X(\mathbf{k}) \to X$, $\mathbf{k}$ integer vector with dim. $h^{1,1}(X)$

Want to know: cohomology dimensions $h^q(X, \mathcal{O}_X(\mathbf{k}))$

Computations can be *very* complicated and may involve Cech cohomology, spectral sequences, Koszul resolutions,...

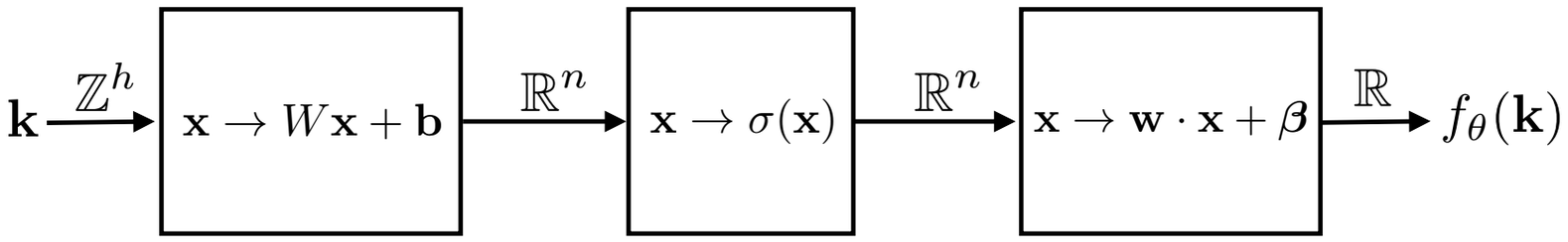
Algorithmic realisations in terms of commutative algebra are often computationally intense.

Example: $X \in \left[\begin{array}{c|cc} \mathbb{P}^1 & 0 & 2 \\ \mathbb{P}^4 & 4 & 1 \end{array} \right]$ $\qquad L = \mathcal{O}_X(-3, 4)$

Simple machine learning of line bundle cohomology

(Fabian Ruehle 1706.07024)

network:



function: $$f_\theta(\mathbf{k}) = \sum_{i=1}^{n} \sum_{j=1}^{h} (w_i \sigma(W_{ij} k_j + b_i) + \beta_i)$$

training data: $\{(\mathbf{k}_i, h^q(\mathcal{O}_X(\mathbf{k}_i))\} \longrightarrow \theta_0$

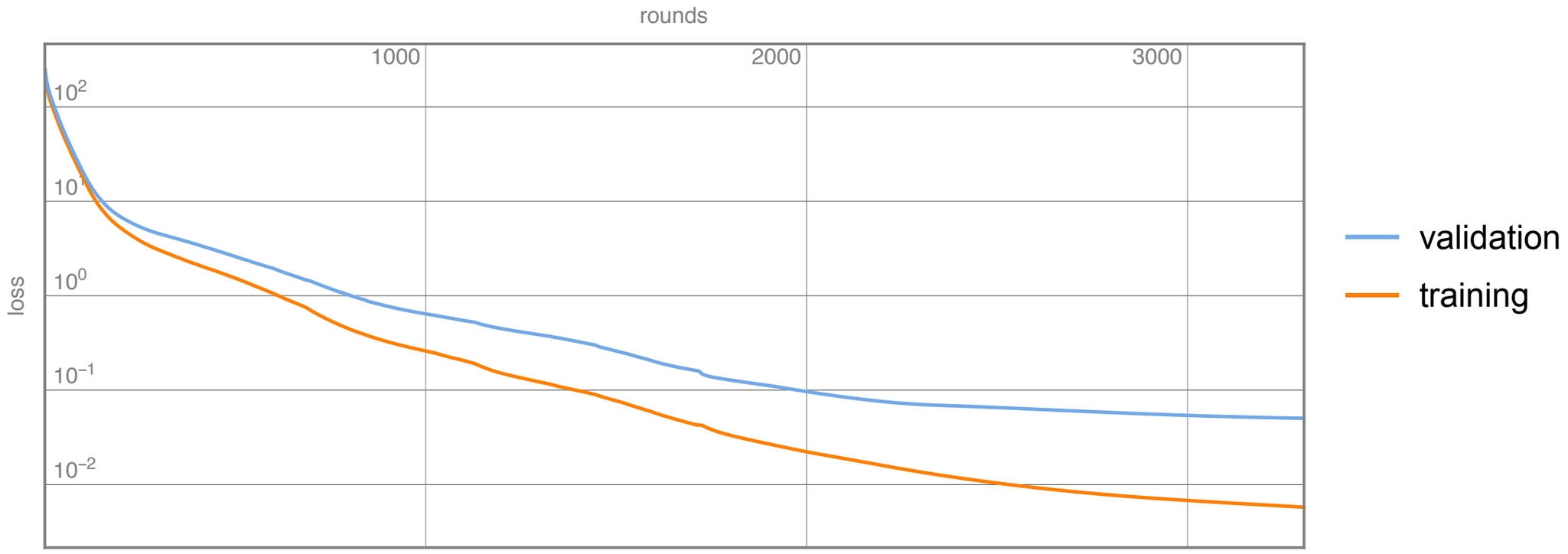
predict cohomology dimensions $h^q(\mathcal{O}_X(\mathbf{k})) \simeq f_{\theta_0}(\mathbf{k})$

Example: line bundle cohomology on dP_2

Want to learn $h^0(dP_2, \mathcal{O}(k_0 l + k_1 e_1 + k_2 e_2))$, $\mathbf{k} = (k_0, k_1, k_2)$

Training data: about 1000 cohomology values from a box $|k_i| \leq 10$

Number of neurons in first layer: 100



Box $|k_i| \leq 10$: net gives correct cohomology for 98% of line bundles

Box $|k_i| \leq 15$: this rate decreases to 73%

This can be repeated, with refinements, for other surfaces and three-folds.

Advantages:

- Fast computation of cohomology dimensions from trained net
- Accurate in 90% of cases, sometimes more

Disadvantages:

- Accurate in only 90% of cases
- Fails outside the ``training box''
- Black box: offers no insight into structure of cohomology

Can we use ML to learn more about the structure of cohomology?

Formulae for line bundle cohomology

(Constantin, AL 1808.09992, Larfors, Schneider 1906.00293)

Complicated computations mask a relatively simple structure. ``Experimental Mathematics" indicates the following:

The Picard lattice splits into regions (often cones). In each region $h^q(X, \mathcal{O}_X(\mathbf{k}))$ is a polynomial in $\mathbf{k}$ with degree equal to the complex dimension of X .

Example 1: $X \in \left[\begin{array}{c|cc} \mathbb{P}^1 & 0 & 2 \\ \mathbb{P}^4 & 4 & 1 \end{array}\right]$

$$h^0(\mathcal{O}_X(\mathbf{k})) = \begin{cases} \mathrm{ind}(\mathcal{O}_X(\mathbf{k})) & k_1 \geq -1 \text{ and } k_2 > 0 \\ \mathrm{ind}(\mathcal{O}_X(\mathbf{k})) + \frac{2}{3}k_1 - \frac{2}{3}k_1^3 & k_1 < -1 \text{ and } k_1 + k_2 > 0 \\ 0 & k_2 < 0 \text{ or } k_1 + k_2 < 0 \\ k_1 + 1 & k_1 \geq 0 \text{ and } k_2 = 0 \\ k_2 + 1 & k_2 > 0 \text{ and } k_1 + k_2 = 0 \end{cases}$$

$$\mathrm{ind}(\mathcal{O}_X(\mathbf{k})) = 2k_1 + \tfrac{14}{3}k_2 + 2kk_1k_2^2 + \tfrac{4}{3}k_2^3.$$

Example 2: dP_1

$$h^0(\mathcal{O}_{\mathrm{dP}_1}(k_0,k_1)) = \begin{cases} \frac{1}{2}(k_0+1)(k_0+2) & k_0 \geq 0\,,\ k_1 \geq 0 \\ \frac{1}{2}(k_0+1)(k_0+2) + \frac{1}{2}k_1(1-k_1) & k_1 < 0\,,\ k_0 > 0\,,\ k_0+k_1 \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

Some understanding for surfaces in terms of index formula:

(Brodie, Constantin, Deen, AL 1906.08769)

$$\tilde{D} = D - \sum_{C \in \mathcal{I}} \theta(-C\cdot D)\,\mathrm{ceil}\left(\frac{C\cdot D}{C^2}\right) C \qquad \Longrightarrow \qquad h^0(\tilde{D}) = h^0(D)$$

In many cases, there is a vanishing theorem (Kodaira or similar) which shows that $h^q(\mathcal{O}(\tilde{D})) = 0$ for $q > 0$. In such cases

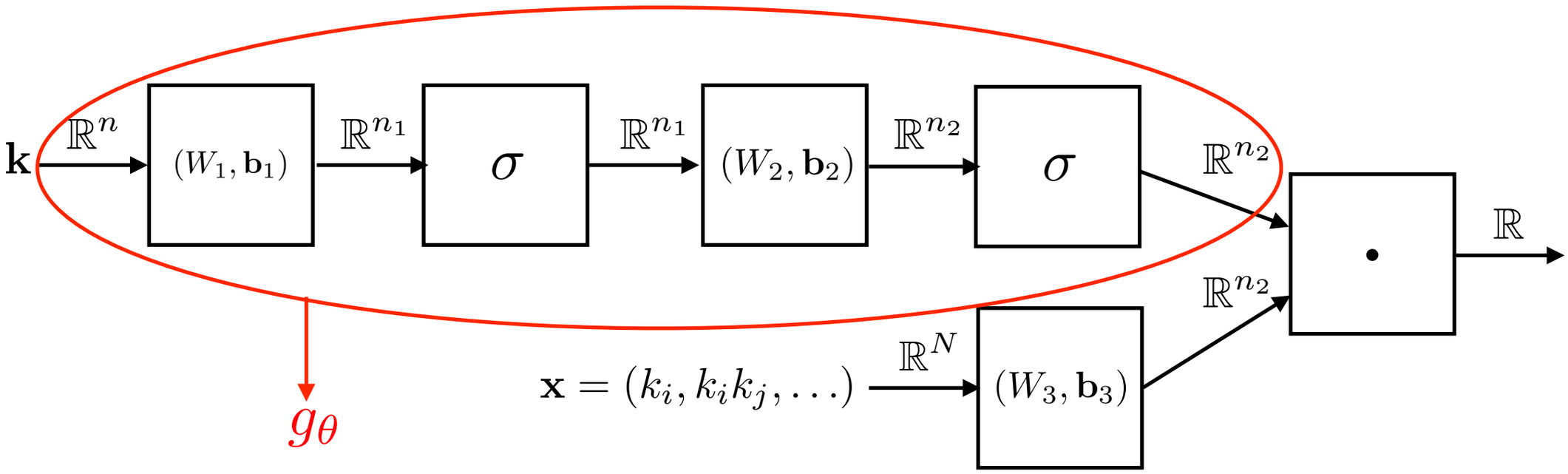
$$h^0(\mathcal{O}(D)) = \mathrm{ind}(\tilde{D})$$

No clear mathematical understanding for three-folds and higher yet.

Can we use ML to conjecture cohomology formulae?

(C. Brodie, A. Constantin, R. Deen, AL, 1906.08769,
D. Klaever, L Schlechter 1809.02547)

Design a net which matches the structure of the formula:



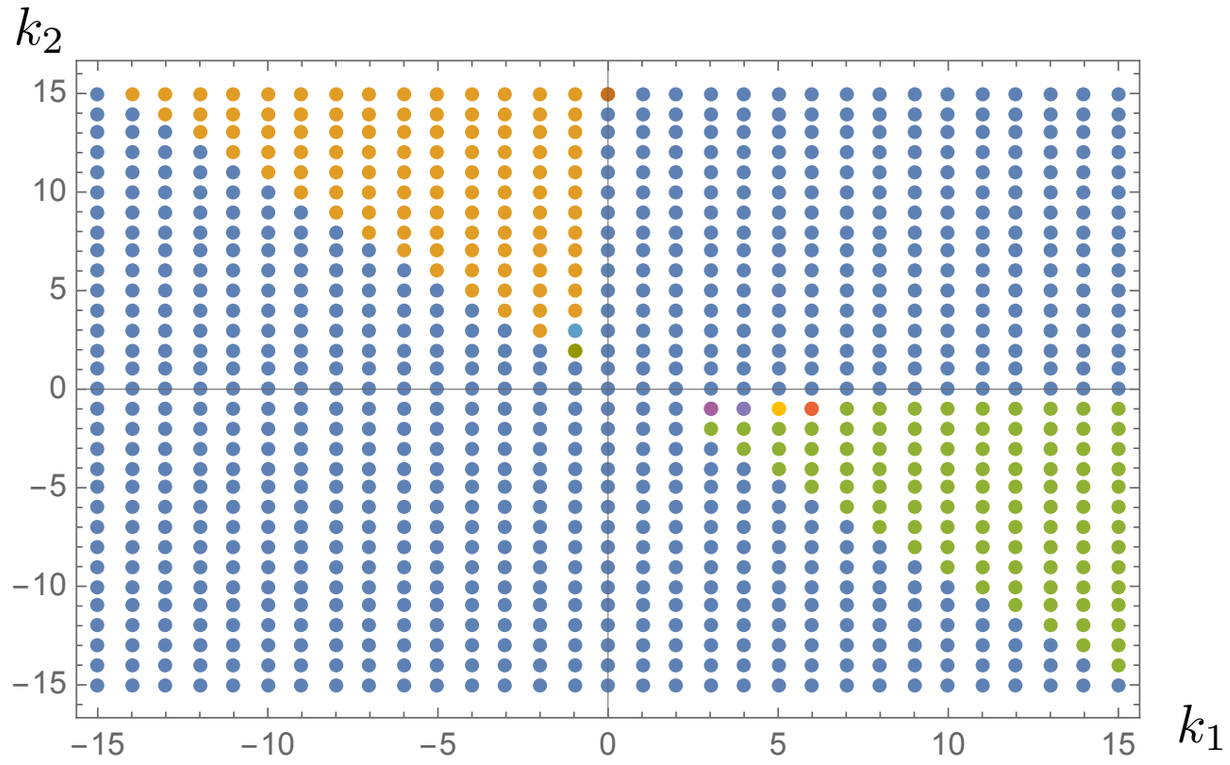
Assume net has been trained: $\rightarrow$ $\quad g_{\theta_0},\ W_{30},\ \mathbf{b}_{30}$

$$a_0 \simeq g_{\theta_0}(\mathbf{k}) \cdot \mathbf{b}_{30} \qquad\qquad \mathbf{a} \simeq g_{\theta_0}(\mathbf{k}) W_{30}$$

Line bundles with similar $(a_0, \mathbf{a})$ are in the same region. This can be used to identify regions and the polynomials.

Example 1: bi-cubic $X \in \left[\begin{array}{c|c} \mathbb{P}^2 & 3 \\ \mathbb{P}^2 & 3 \end{array}\right]$, $\quad h^1(X, \mathcal{O}_X(k_1, k_2))$

1) Train and identify regions:

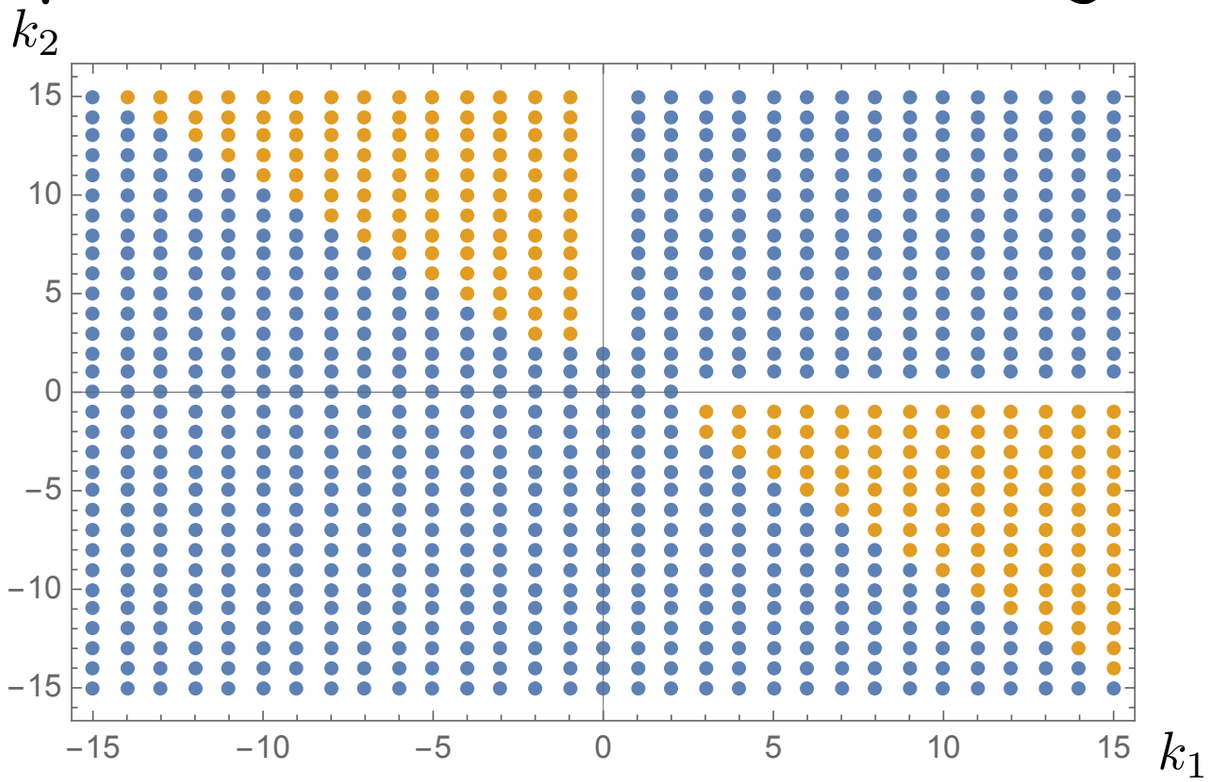


2) Find correct cubic polynomial for each region by a fit:

blue: $h^1(\mathcal{O}_X(k_1, k_2)) = 0$

yellow/green: $h^1(\mathcal{O}_X(k_1, k_2)) = -\frac{3}{2}(k_1 + k_2)(2 + k_1 k_2)$

3) Use these equations to find the exact regions:



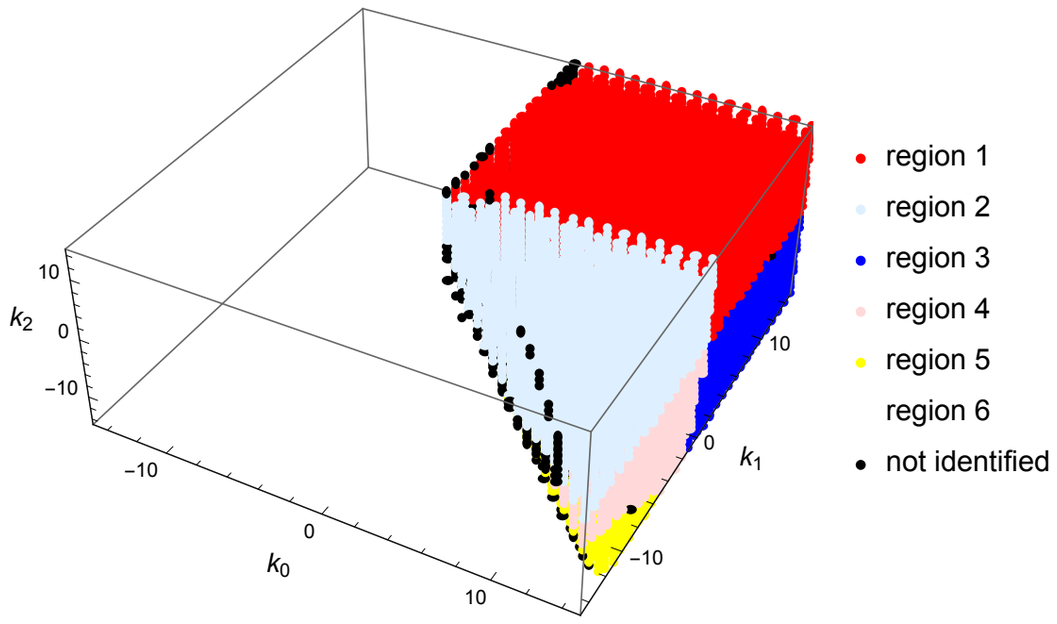
4) Find equations for boundaries of regions

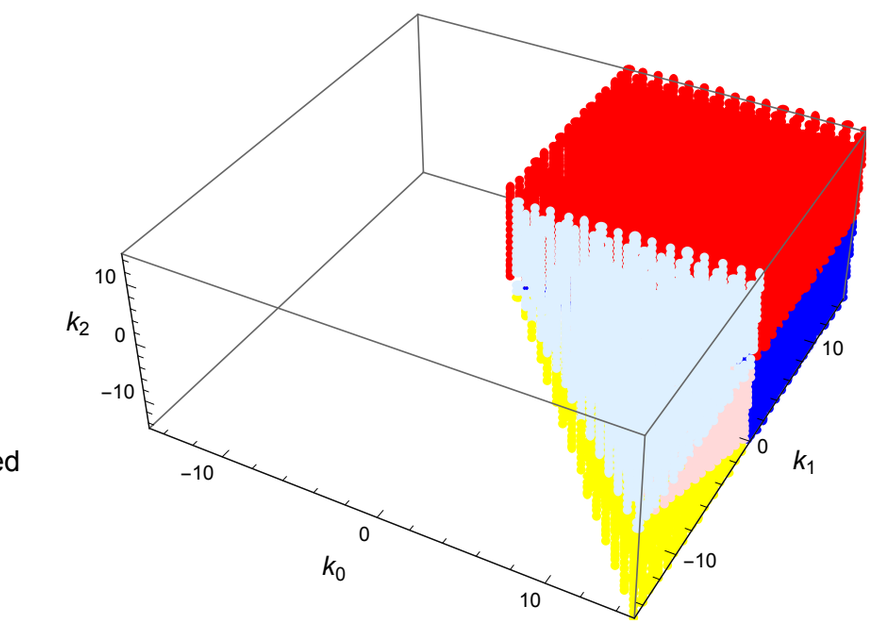
$$h^1(\mathcal{O}_X(\mathbf{k})) = \begin{cases} \frac{1}{2}(-1+k_2)(-2+k_2)\ , & k_1 = 0,\ k_2 > 0 \\ -\mathrm{ind}(\mathcal{O}_X(\mathbf{k}))\ , & k_1 < 0,\ k_2 > -k_1 \\ 0 & \text{otherwise}\ , \end{cases}$$

$$\mathrm{ind}(\mathcal{O}_X(\mathbf{k})) = \tfrac{3}{2}(k_1+k_2)(2+k_1k_2)$$

Can be shown from Cech cohomology/Koszul sequence.

Example 2: dP_2 , $h^0(\mathcal{O}_{dP_2}(k_0 l + k_1 e_1 + k_2 e_2))$





$$h^0(\mathcal{O}_{\mathrm{dP}_2}(\mathbf{k})) = \begin{cases} 1 + \frac{3}{2}k_0 + \frac{1}{2}k_0^2 + \frac{1}{2}k_1 - \frac{1}{2}k_1^2 + \frac{1}{2}k_2 - \frac{1}{2}k_2^2 & \text{in region } 1\,, \\ 1 + 2k_0 + k_0^2 + k_1 + k_0k_1 + k_2 + k_0k_2 + k_1k_2 & \text{in region } 2\,, \\ 1 + \frac{3}{2}k_0 + \frac{1}{2}k_0^2 + \frac{1}{2}k_2 - \frac{1}{2}k_2^2 & \text{in region } 3\,, \\ 1 + \frac{3}{2}k_0 + \frac{1}{2}k_0^2 + \frac{1}{2}k_1 - \frac{1}{2}k_1^2 & \text{in region } 4\,, \\ 1 + \frac{3}{2}k_0 + \frac{1}{2}k_0^2 & \text{in region } 5\,. \\ 0 & \text{in region } 6\,. \end{cases}$$

Can be shown from index formula.

Region 1:	$-k_1 \geq 0$	$-k_2 \geq 0$	$k_0 + k_1 + k_2 \geq 0$			
Region 2:			$k_0 + k_1 + k_2 < 0$	$k_0 + k_1 \geq 0$	$k_0 + k_2 \geq 0$	
Region 3:	$-k_1 < 0$	$-k_2 \geq 0$			$k_0 + k_2 \geq 0$	
Region 4:	$-k_1 \geq 0$	$-k_2 < 0$			$k_0 + k_2 \geq 0$	
Region 5:	$-k_1 < 0$	$-k_2 < 0$				$k_0 \geq 0$
Region 6:	otherwise					

ML can be used to generate conjectures for cohomology formulae.

Can ML help with the large amount of string data?

Heterotic line bundle standard models

(Anderson, Constantin, Gray, Lukas, Palti, 1106.4804, 1202.1757, 1307.4787)

Based on line bundle sums over CY three-folds:

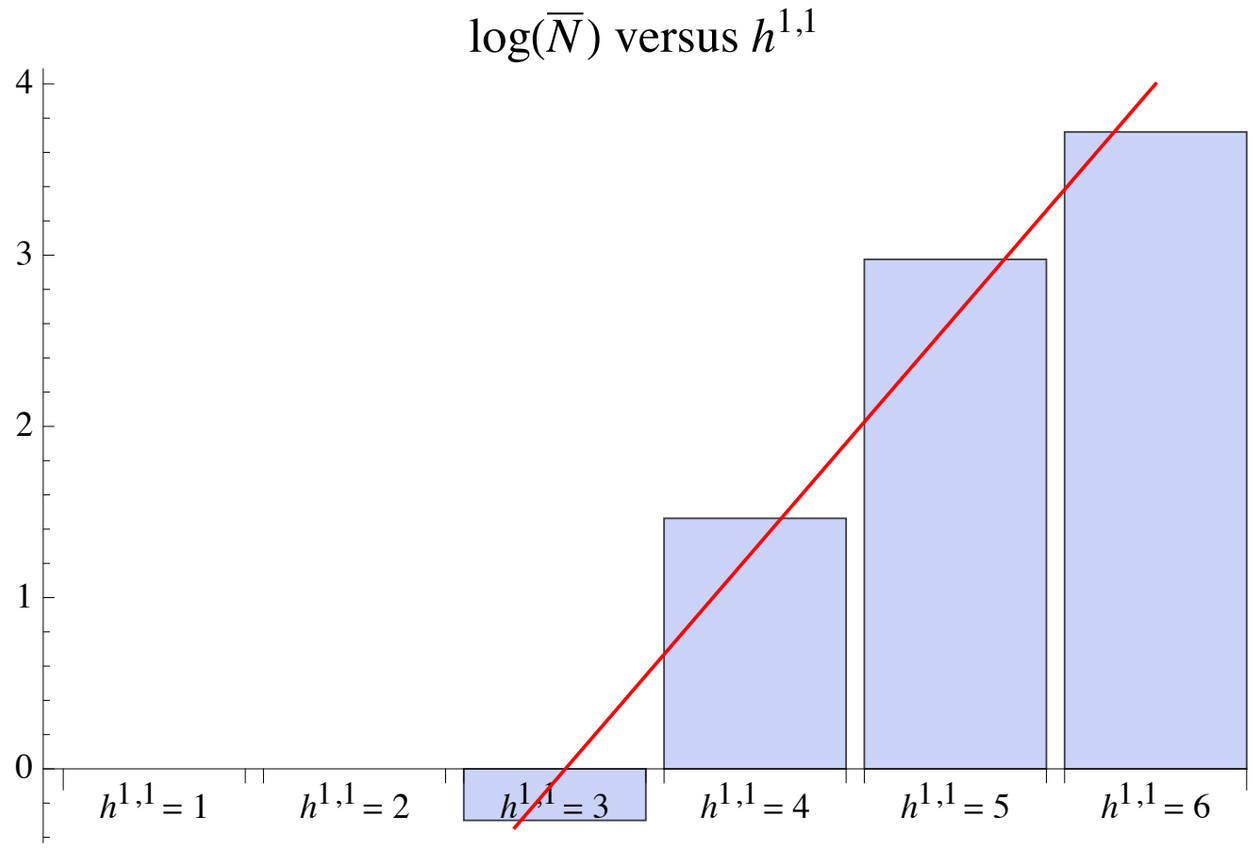
$$V = \bigoplus_{a=1}^{5} \mathcal{O}_X(\mathbf{k}_a) \longrightarrow X$$

Specified by three-fold X and $h^{1,1}(X) \times 5$ integer matrix $K = (k_a^i)$.

Number of models per CY: $N_{\text{tot}} \sim 10^{5h^{1,1}(X)}$

A ``brut-force" scan for $h^{1,1}(X) \leq 6$ shows a tiny fraction of these are quasi-standard models: *

* A consistent string model with the right gauge group and 3 chiral families



$$\log(\bar{N}(h)) \simeq -4.1 + 1.4\,h$$

Bold extrapolation leads to:

For CICYs: $h_{\max} = 19$ $N(h_{\max}) \simeq 10^{23}$

All known CYs: $h_{\max} = 491$ $N(h_{\max}) \simeq 10^{662}$

There is no way to check this by systematic scanning.

Can ML help?

Supervised learning of standard models

Example: $X \in \left[\begin{array}{c|ccc} \mathbb{P}^1 & 0 & 1 & 1 \\ \mathbb{P}^1 & 0 & 1 & 1 \\ \mathbb{P}^1 & 1 & 1 & 0 \\ \mathbb{P}^1 & 1 & 1 & 0 \\ \mathbb{P}^1 & 1 & 0 & 1 \\ \mathbb{P}^1 & 1 & 0 & 1 \end{array}\right]$, ~ 17000 SMs, same number of non-SMs

Training set: $\{K \to 0 \text{ or } 1\}$

non-SM SM

Two examples from the data set:

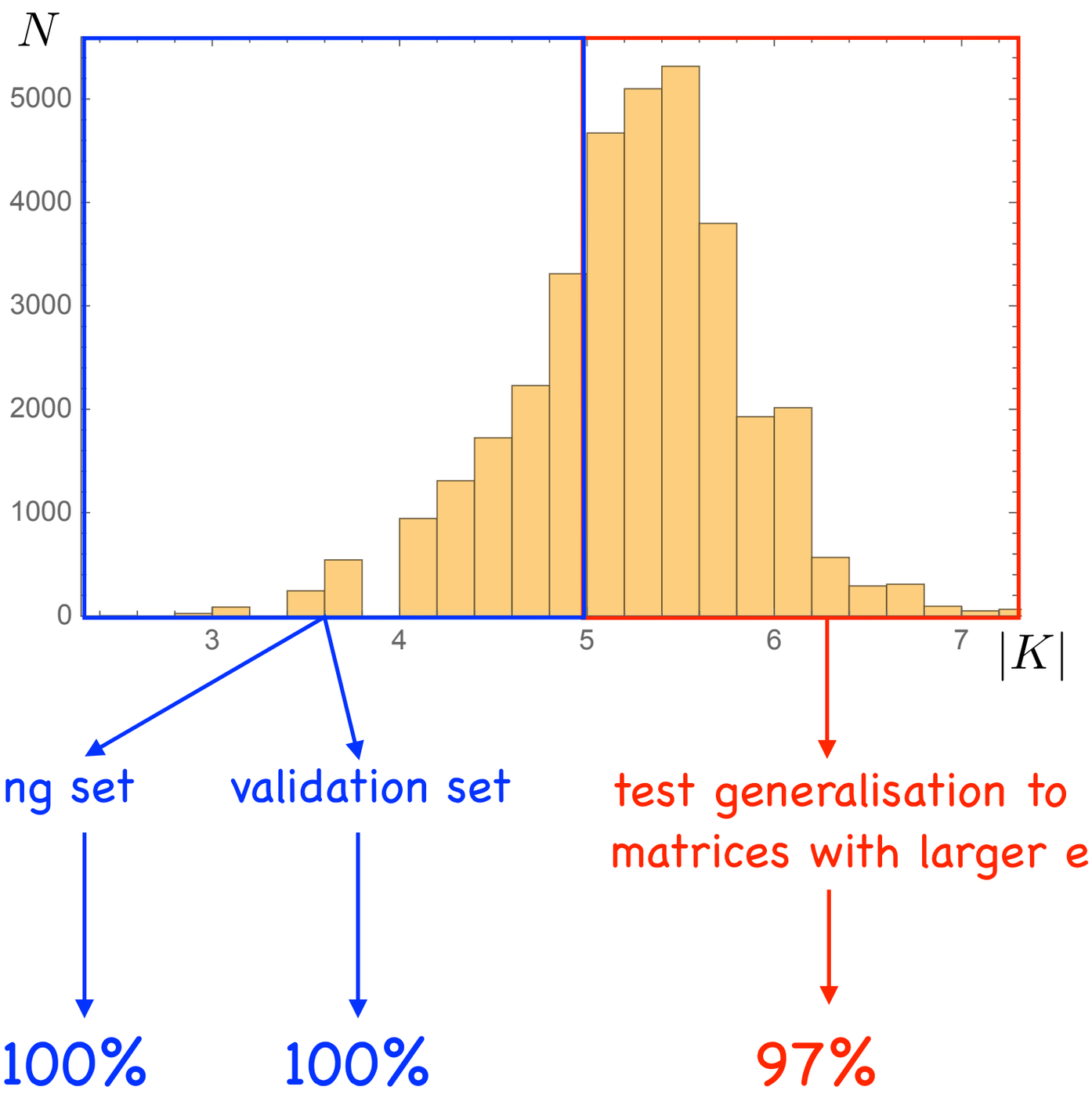
SM

$$K = \begin{pmatrix} -1 & -1 & -1 & 1 & 2 \\ 0 & -2 & 0 & 1 & 1 \\ -1 & 1 & -1 & 0 & 1 \\ 1 & 0 & 1 & 0 & -2 \\ 0 & 1 & 0 & 0 & -1 \\ 1 & 0 & 1 & -2 & 0 \end{pmatrix} \to 1$$

non-SM

$$K = \begin{pmatrix} 2 & -1 & -1 & 0 & 0 \\ 0 & 1 & 0 & -1 & 0 \\ -1 & 2 & 2 & -1 & -2 \\ 1 & 0 & 0 & 0 & -1 \\ -1 & 0 & 1 & 1 & -1 \\ 1 & -1 & 0 & -1 & 1 \end{pmatrix} \to 0$$

Network: simple 2 or 3 layer, fully connected



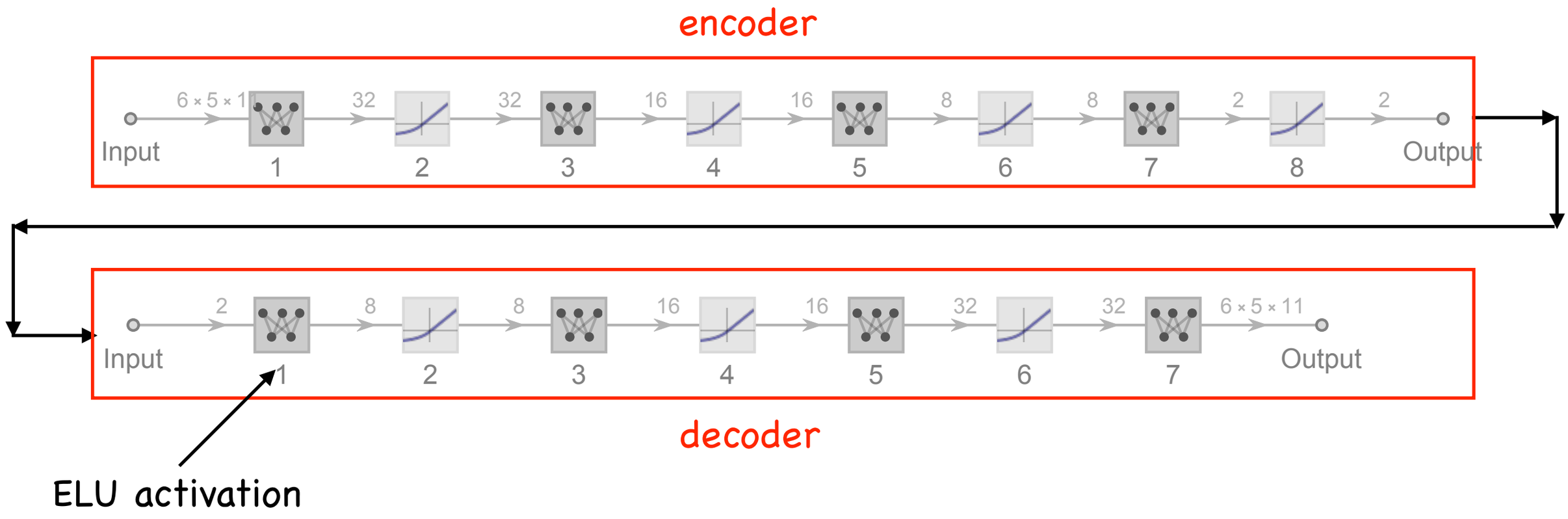
This provides a fast method to distinguish SM and non-SMs which works beyond the training range.
Still requires testing every matrix -> limited improvement.

Auto-encoding standard models

Use the same training set of SMs and non-SMs, one-hot encoded.

(Vaudrevange 1811.05993)

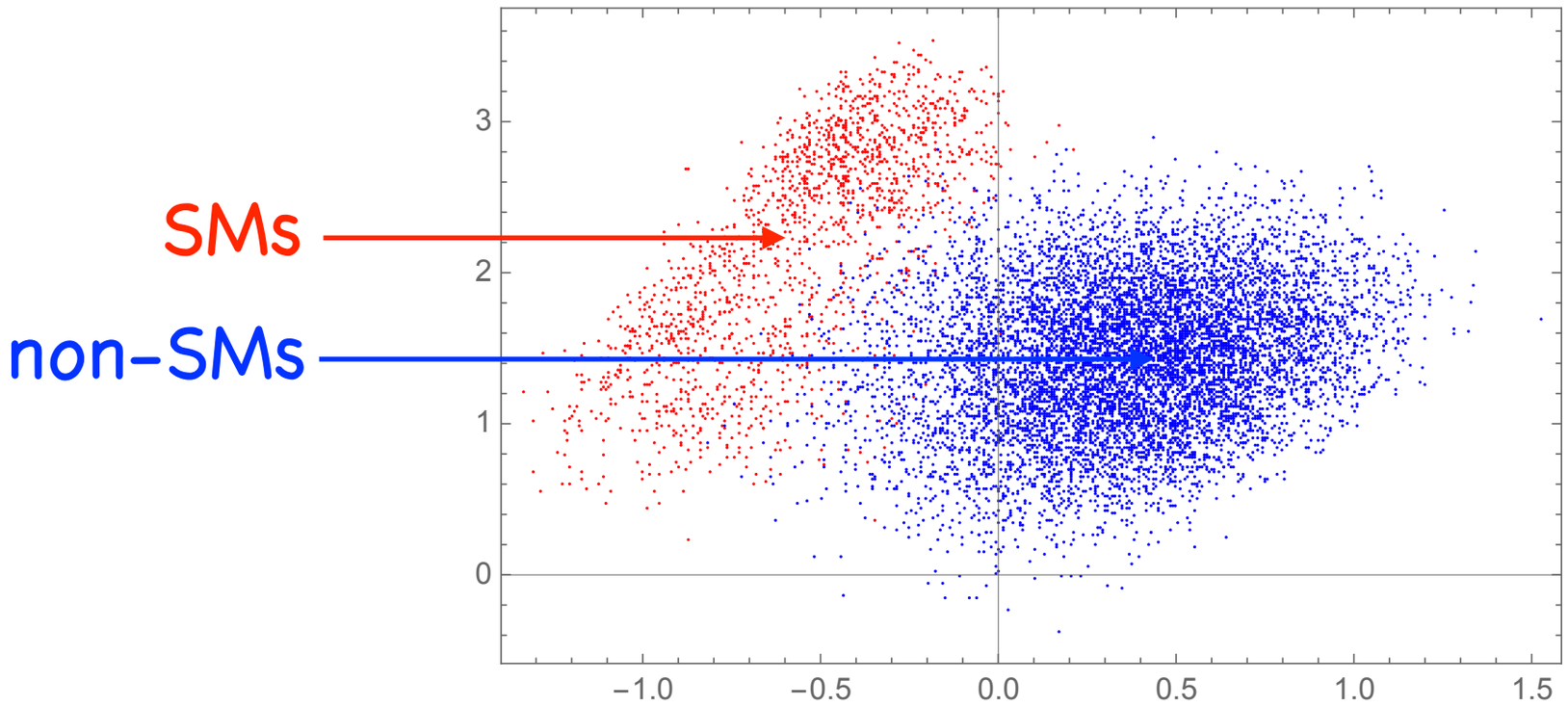
Network:



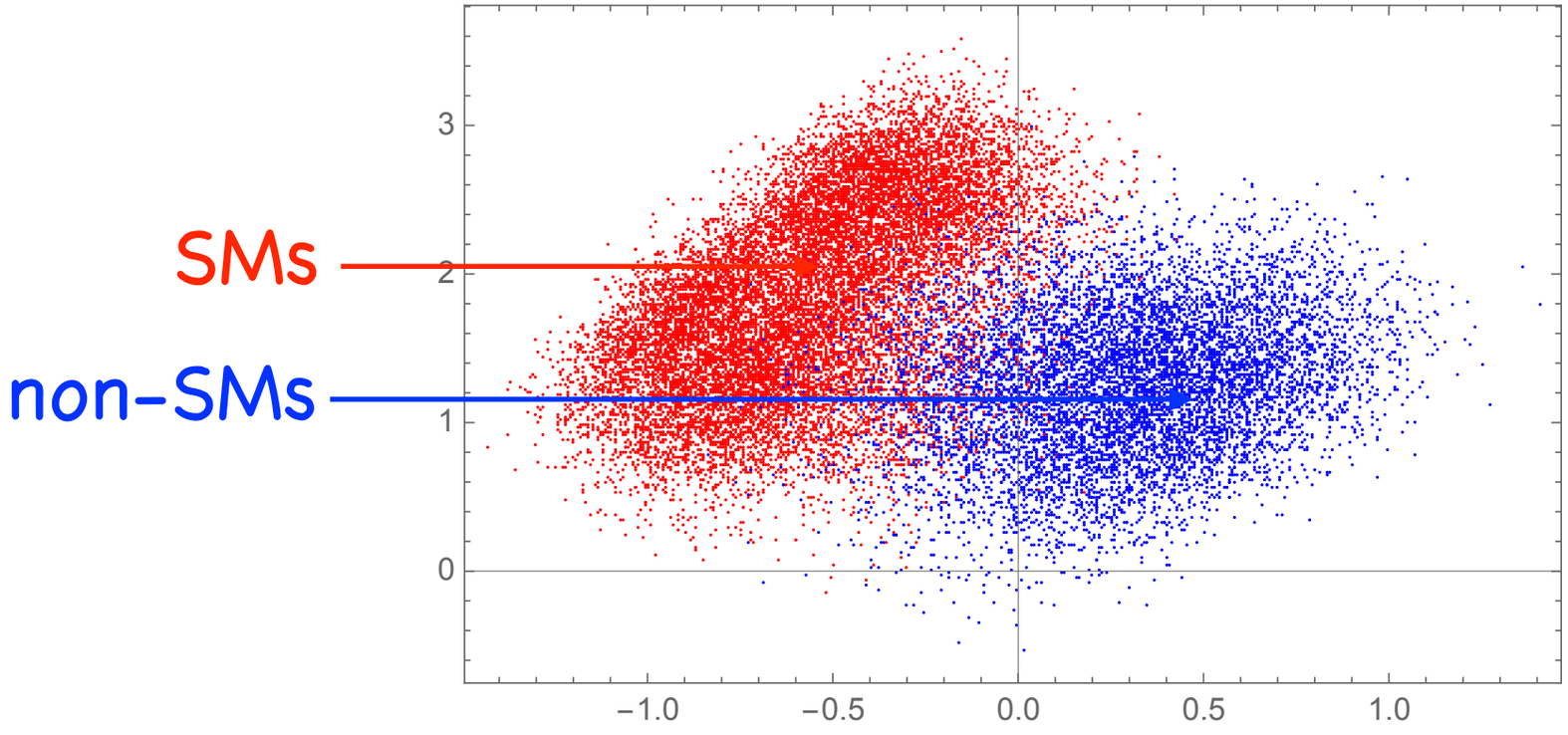
Training set: SM and non-SMs K for $|K| \leq 5$

Training: minimise $|K_{\text{in}} - K_{\text{out}}|$

Use trained encoder to map into 2d space:



$$|K| \leq 5$$



$$|K| > 5$$

Auto-encoder can distinguish SMs and non-SMs and generalises beyond training range.

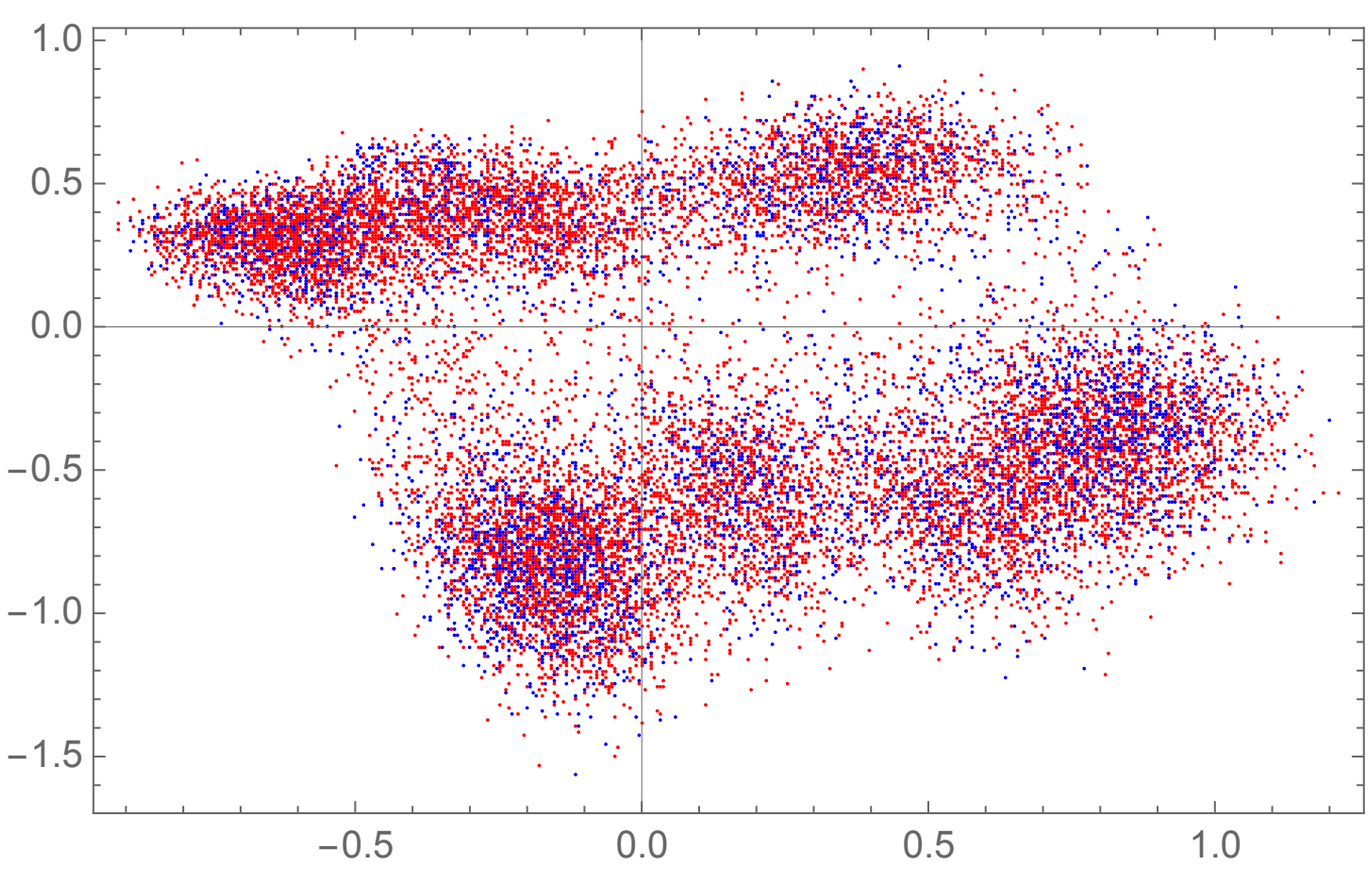
Standard-like models with and without Higgs

Training set: $\{K \to 0 \text{ or } 1\}$

without Higgs with Higgs

It is not straightforward to get a network to learn the difference.

E.g. auto-encoder:



blue: without Higgs

red: with Higgs

We need a feature-enhanced data set:

$$K = (k_a^i) \quad \longrightarrow \quad \tilde{K} = (k_a^i, k_a^i k_a^j, k_a^i k_a^k k_a^m)$$

This is not unexpected, given the line bundle cohomology results.

Training set: $\{\tilde{K} \to 0 \text{ or } 1\}$

without Higgs

with Higgs

A simple fully connected net (3 layers, width (256,64,16)) leads to a 100% success rate.

First steps into RL

(J. Halverson, B. Nelson, F. Ruehle 10903.11616)
(M. Larfors, R. Schneider, 2003.04817)

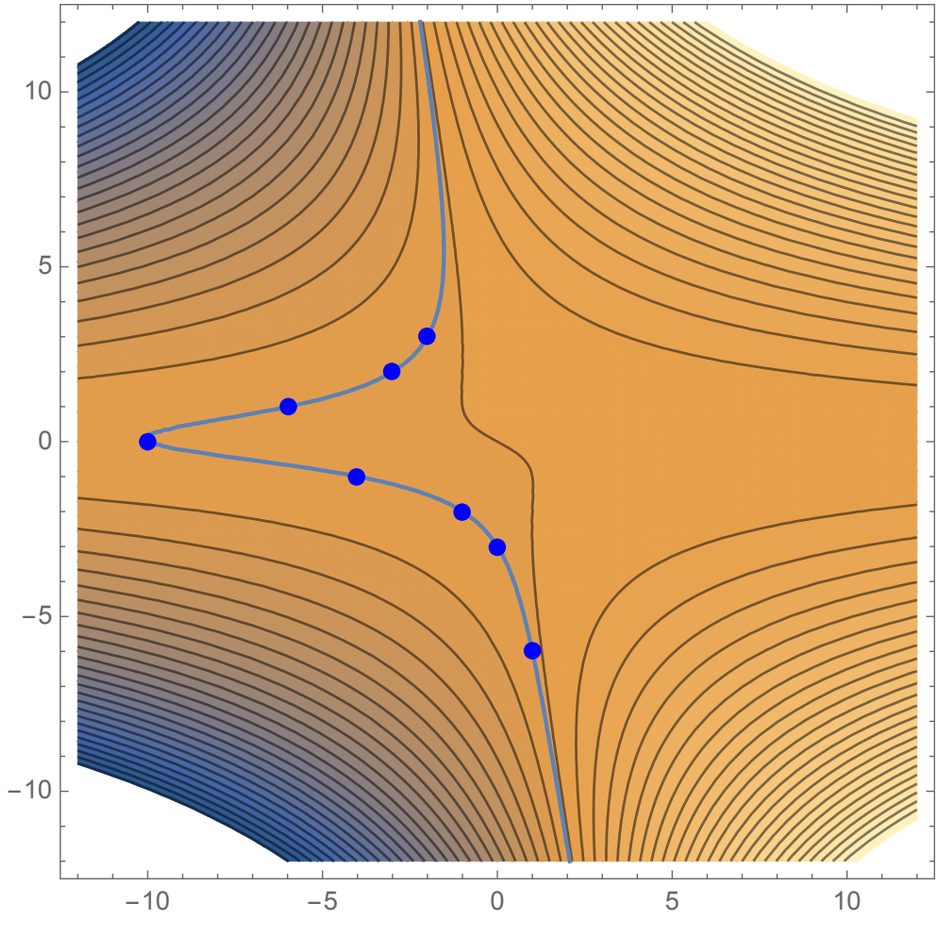
Environment: line bundles $L = \mathcal{O}_X(\mathbf{k}) \to X$ over given CY X

actions: $k_i \to k_i \pm 1$ for a given i

goal: find line bundles $\mathcal{O}_X(\mathbf{k})$ with a given ``target index" I, so

$$\mathrm{ind}(\mathcal{O}_X(\mathbf{k})) \stackrel{!}{=} I$$

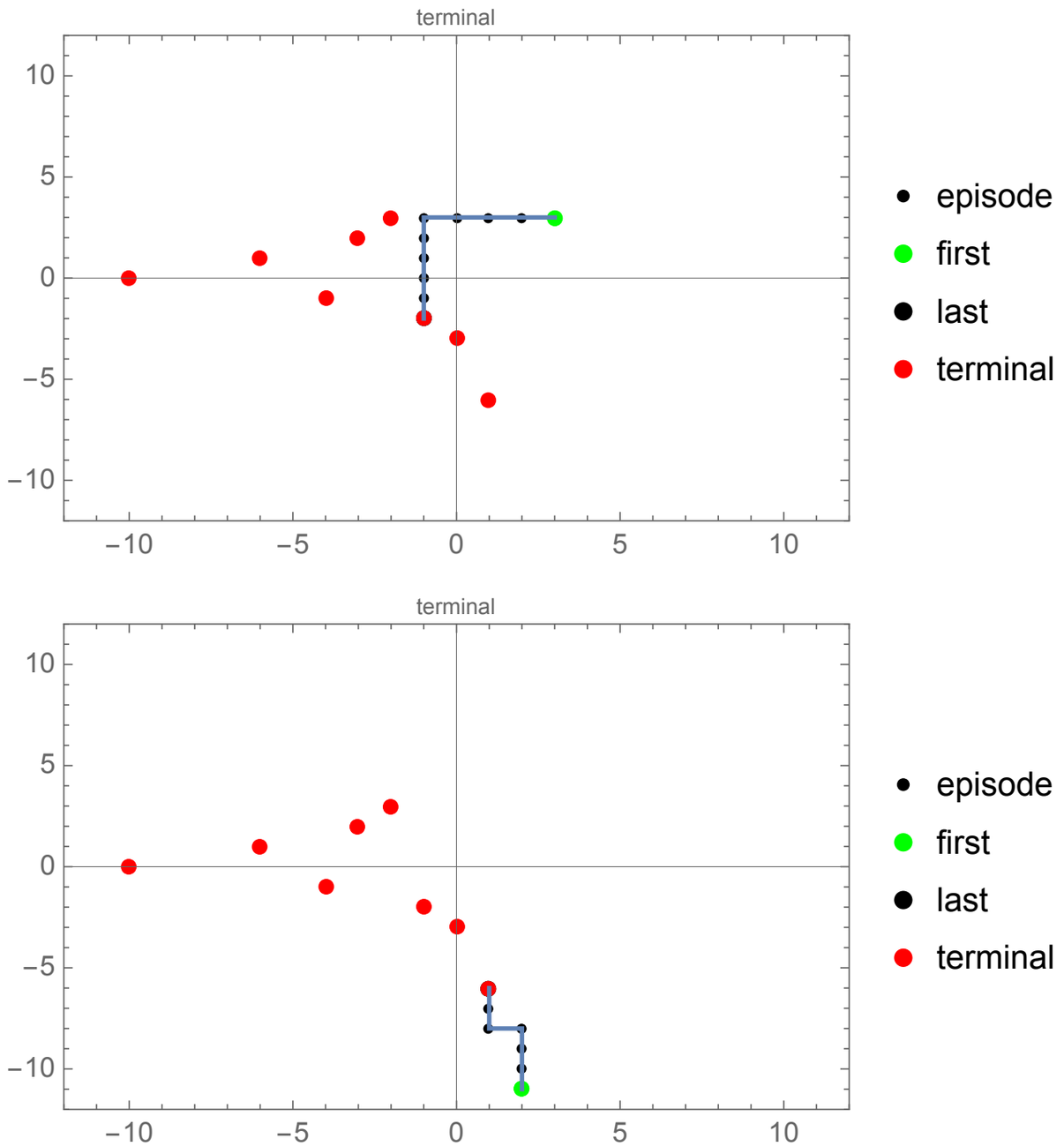
Example: bi-cubic CY, $L = \mathcal{O}_X(k_1, k_2)$, demand $\mathrm{ind}(L) = -20$

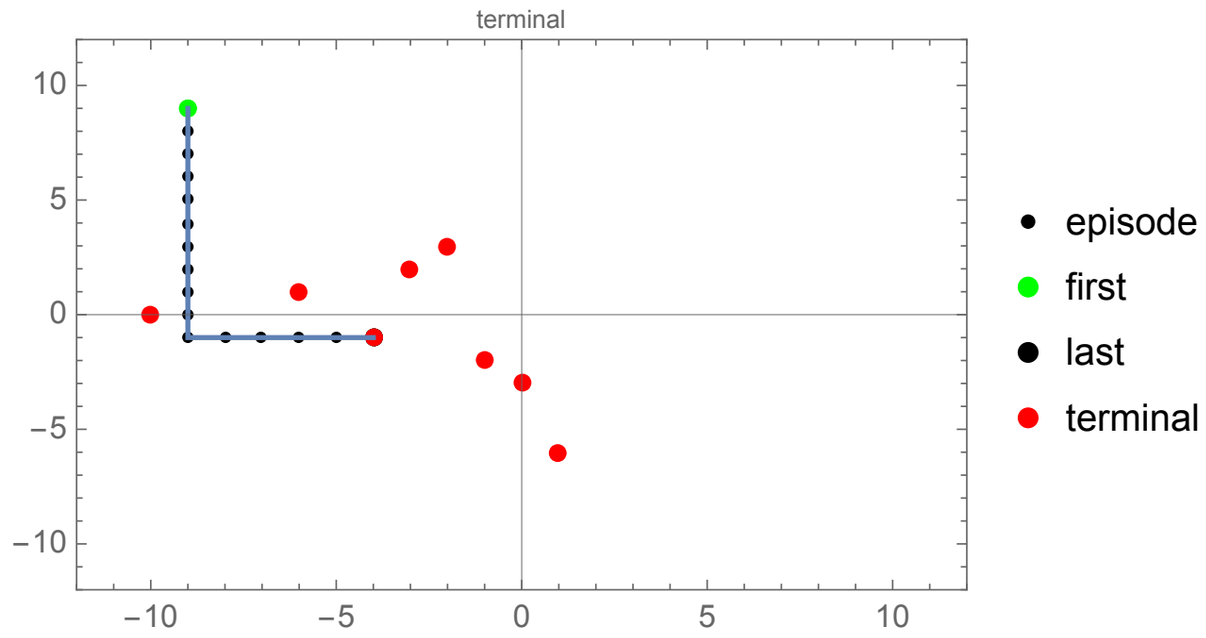


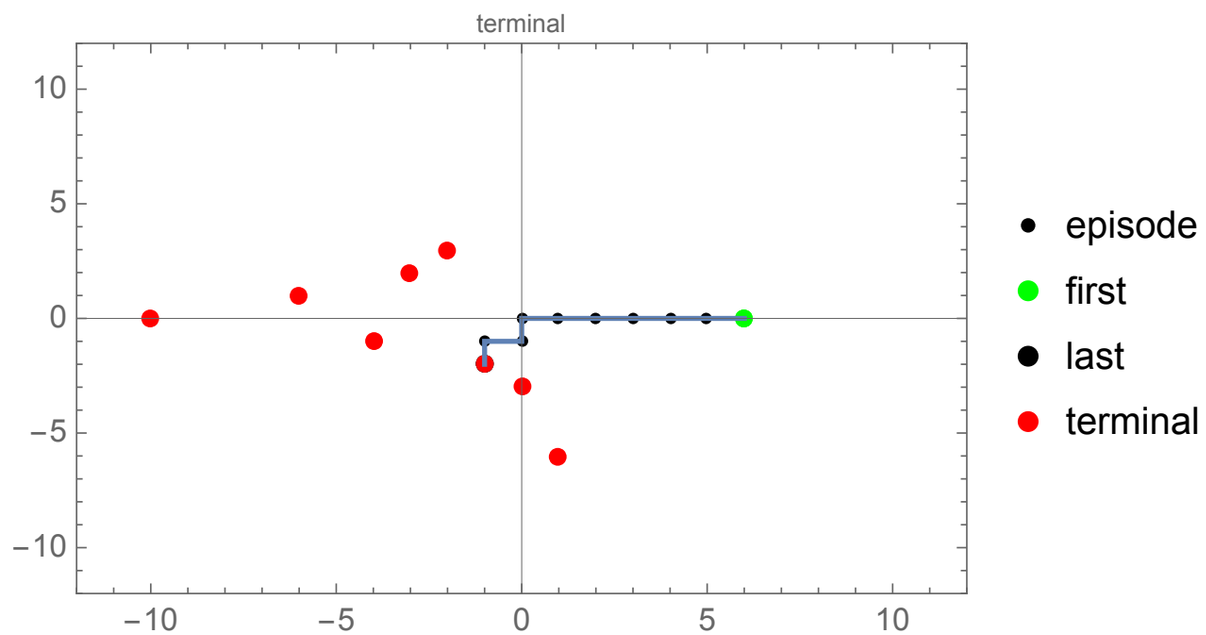
Method: REINFORCE, training box, $|k_i| \leq 12$, 4 layers, width 32
realised in Mathematica

Trained policy net guides to states with the correct index for any random initial state!

Some sample trajectories:







Conclusions

- ML can be used to generate non-trivial conjectures for line bundle cohomology formulae.

- Simple fully connected networks with supervised training can be used to distinguish SMs and non-SMs and they generalise beyond the training range.

- SM and non-SMs can also be distinguished with unsupervised learning using auto-encoders.

- This may help to explore more of the string landscape - but reinforcement learning will probably be required.

- Learning non-topological properties such as presence/absence of Higgs is more difficult but can work with feature-engineering.

- RL of topological properties looks promising. Goal is to apply it to non-trivial bundles (e.g. monads).

Thanks