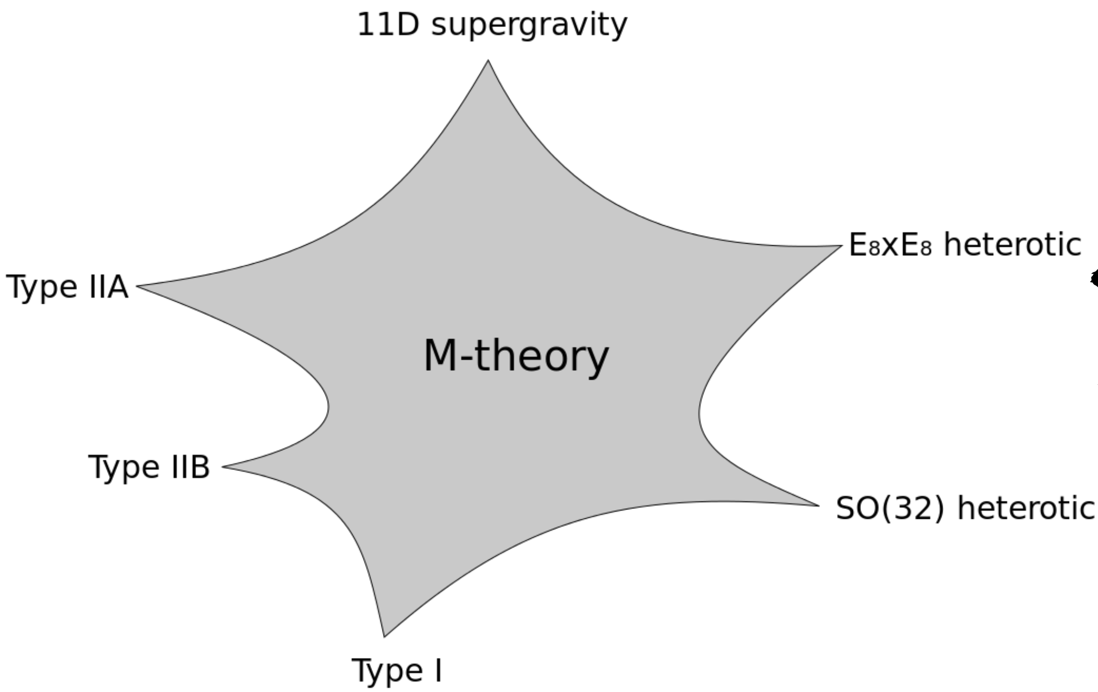
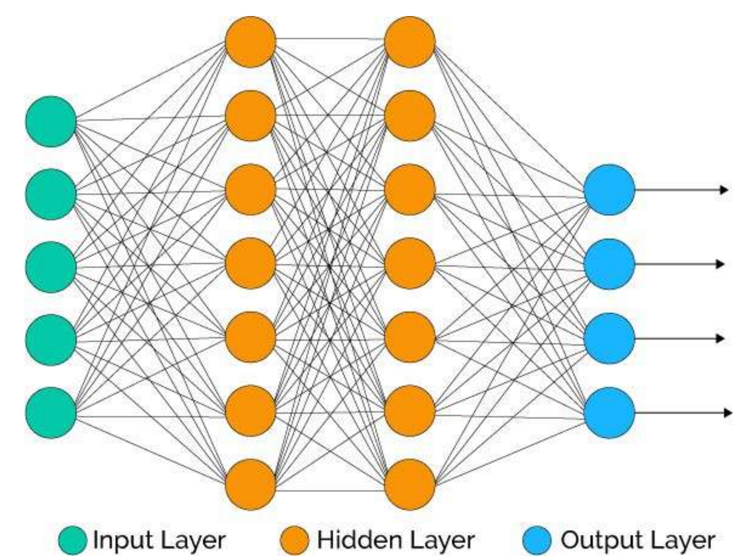
Machine Learning and String Theory

Andre Lukas

Rudolf-Peierls-Centre for Theoretical Physics
University of Oxford

Saturday morning of Theoretical Physics, 29 February 2020





Outline

- AI and strings: History and motivation
- Machine learning basics
- String theory basics
- Machine-learning string theory
- Conclusion

AI and strings: History and motivation

A short history of about three years, starting in Oxford TP . . .

Fabian Ruehle

Sven Krippendorf

Yang-Hui He

Fabian Ruehle: arXiv:1706.07024

``Evolving neural networks with genetic algorithms to study the string landscape

Yang-Hui He: arXiv:1706.02714

``Deep learning the landscape"

A burst of activity since - but still in its infancy . . .

The ``obvious" motivation . . .

- String theory leads to very large data sets, very different from the ``usual" data (pictures, videos,...).

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

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

Can machine learning help uncover features of string data?

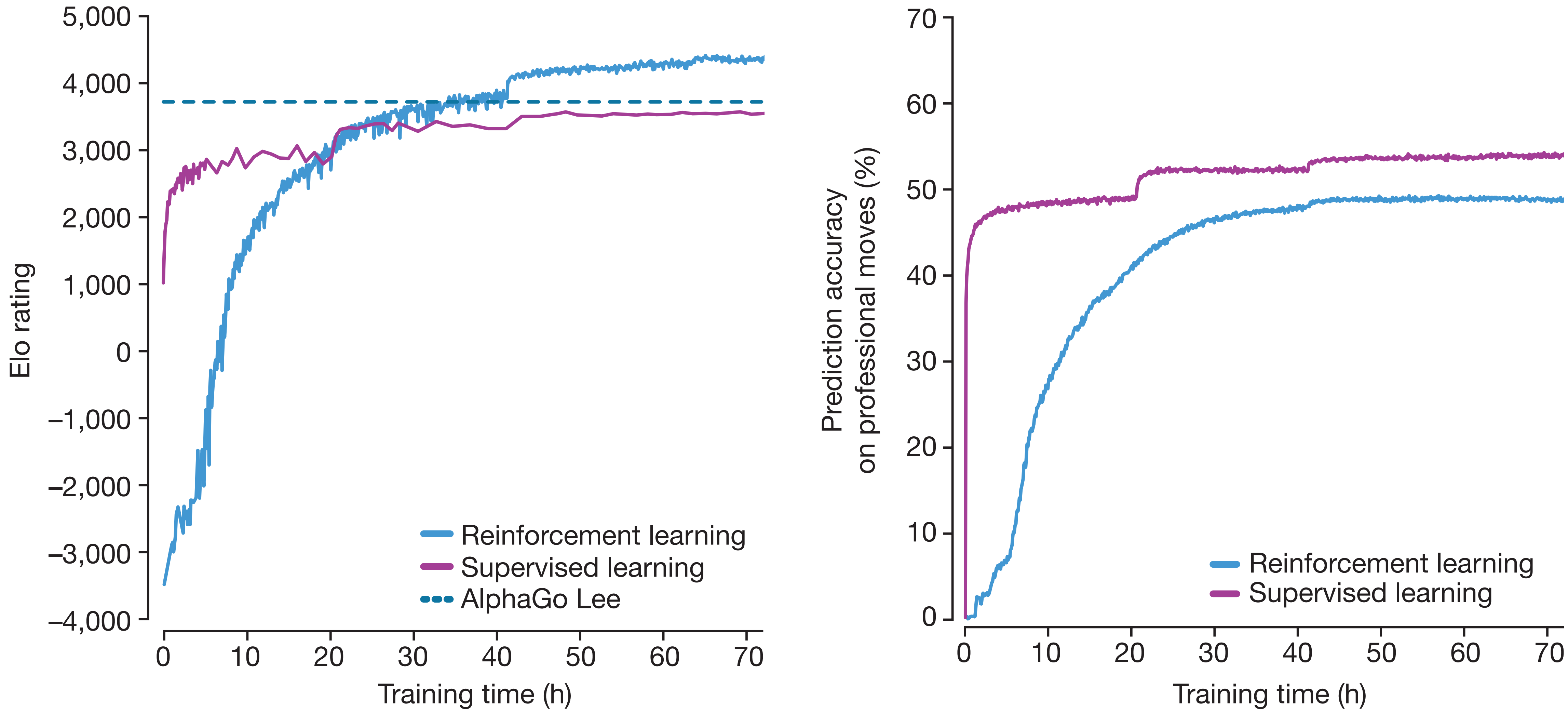
Perhaps machine learning can do more . . .

Example: Game of go ($\sim 10^{800}$ games)

(For comparison: $\sim 10^{40}$ ``sensible" chess games)

Tackling the game of go with machine learning:

(Silver et al, DeepMind, Nature 2017)



. . . and help reveal mathematical structures.

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, in preparation)

Machine learning basics

Simplest approach: supervised learning

Structure of neural network:

$$\mathbb{R}^n \ni \mathbf{x} \longrightarrow \boxed{f_\theta : \mathbb{R}^n \to \mathbb{R}^m} \longrightarrow f_\theta(\mathbf{x}) \in \mathbb{R}^m$$

trainable parameters

Training:

training set: $\{(\mathbf{x}_i, \mathbf{y}_i) \in \mathbb{R}^n \times \mathbb{R}^m\}$

training: minimise loss $L(\theta) = \dfrac{1}{N} \sum_{i=1}^{N} |f_\theta(\mathbf{x}_i) - \mathbf{y}_i|^2 \quad \longrightarrow \quad \theta_0$

Validate and test: Compute $L(\theta_0)$ for unseen data $(\mathbf{x}_i, \mathbf{y}_i)$

Predictions: $\mathbf{x} \to f_{\theta_0}(\mathbf{x})$

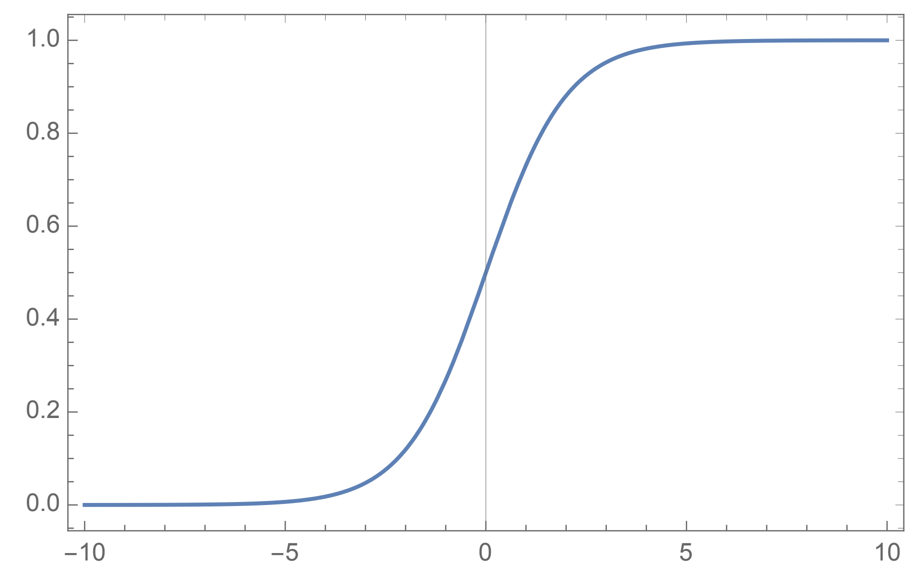
A basic building block: the perceptron

$$\mathbb{R}^n \ni \mathbf{x} \longrightarrow \boxed{\mathbf{x} \to \mathbf{w}\cdot\mathbf{x} + b} \xrightarrow{z} \boxed{z \to \sigma(z)} \longrightarrow f_{(\mathbf{w},b)}(\mathbf{x}) = \sigma(\mathbf{w}\cdot\mathbf{x}+b) \in \mathbb{R}$$

$\mathbf{w}$: weights $\qquad b$: bias

θ

σ : activation function



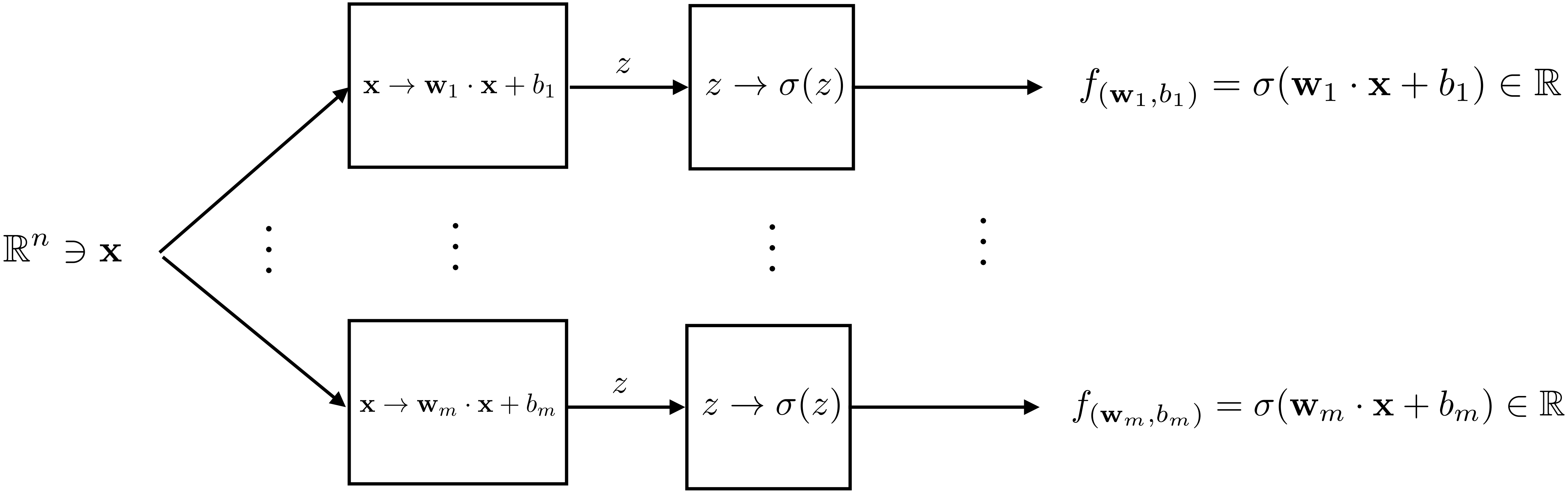
Typically: $\sigma(z) = \dfrac{1}{1+e^{-z}}$

Perceptron relates to the (hyper)plane $\{\mathbf{x} \in \mathbb{R}^n \,|\, \mathbf{w}\cdot\mathbf{x} + b = 0\}$

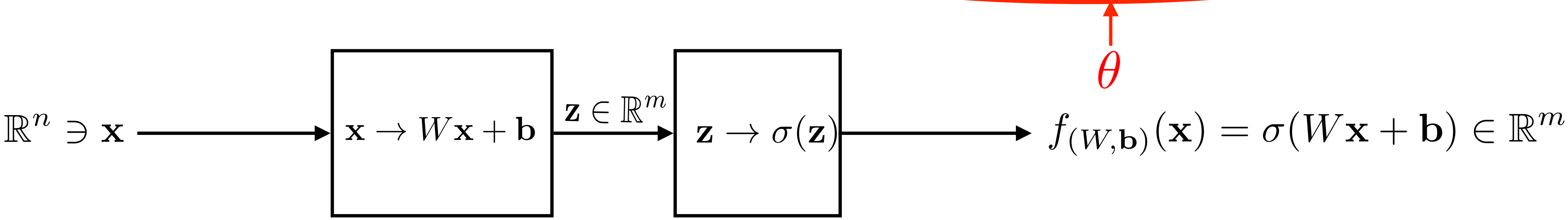
$$f_{\mathbf{w},b}(\mathbf{x}) \simeq \begin{cases} 1 & \mathbf{w}\cdot\mathbf{x}+b > 0 \quad \text{``above plane''} \\ 0 & \mathbf{w}\cdot\mathbf{x}+b < 0 \quad \text{``below plane''} \end{cases}$$

-> Example

The next step: m perceptrons in parallel

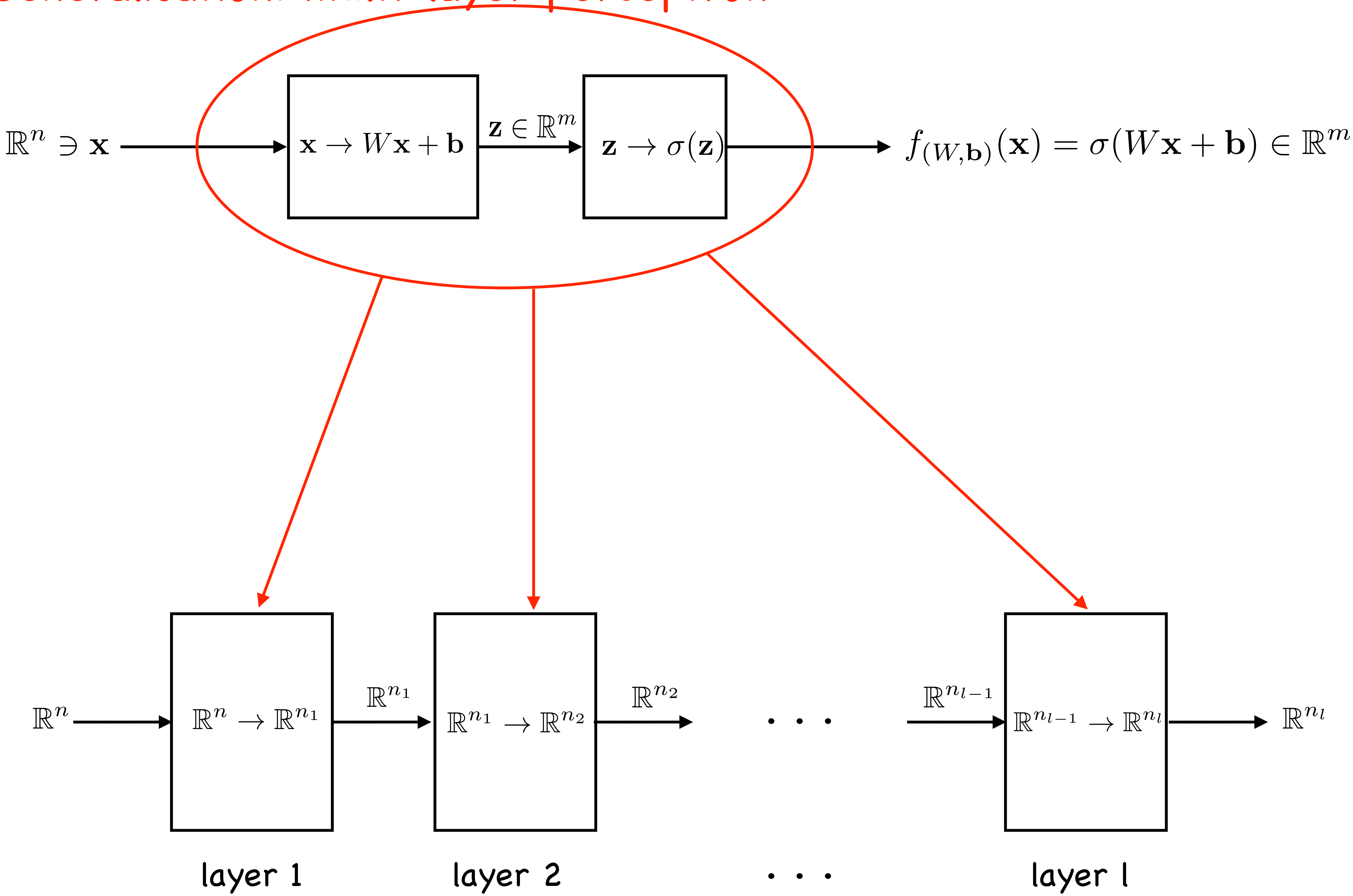


Introduce weight matrix $W = \begin{pmatrix} \mathbf{w}_1 \\ \vdots \\ \mathbf{w}_m \end{pmatrix}$ and bias vector $\mathbf{b} = \begin{pmatrix} b_1 \\ \vdots \\ b_m \end{pmatrix}$



Learns position of m hyperplanes -> pattern recognition

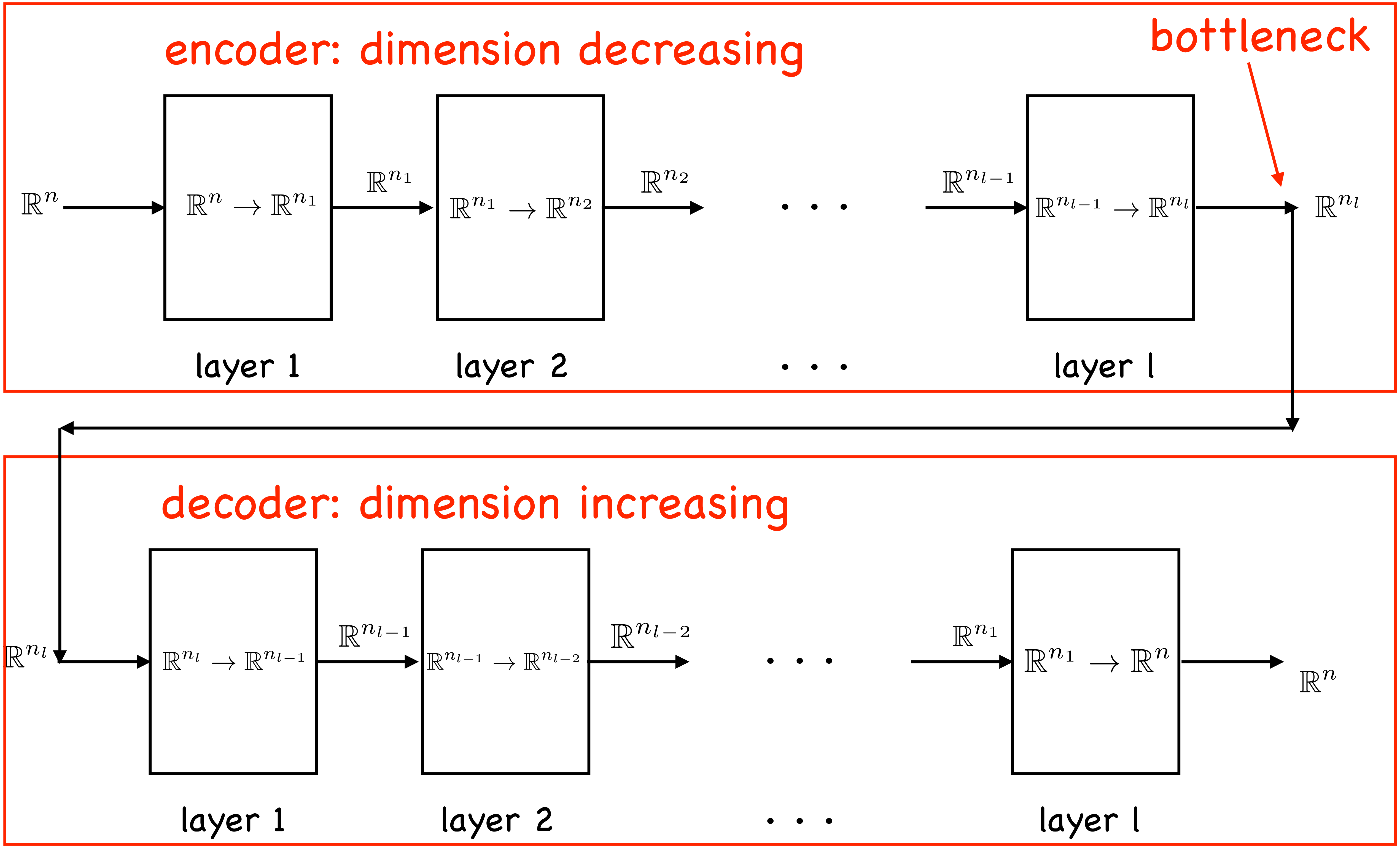
Generalisation: multi-layer perceptron



Many more generalisations of this . . .

-> Example

Unsupervised learning: example auto-encoder



Training set: $\{\mathbf{x}_i\}$

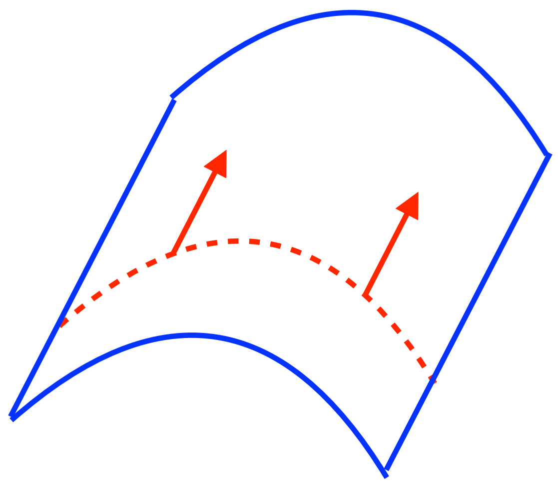
Training: minimise loss $L(\theta) = \frac{1}{N}\sum_{i=1}^{N} |\mathbf{x}_i - f_\theta(\mathbf{x}_i)|^2$

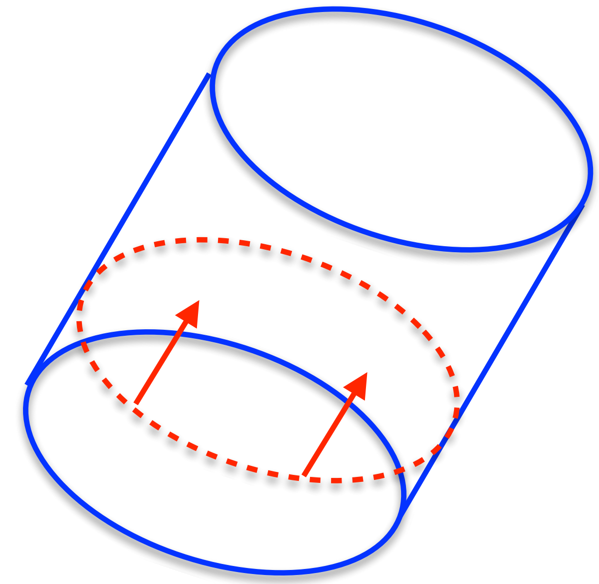
-> Example

String theory basics

String theory recap

- open string   closed string





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

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

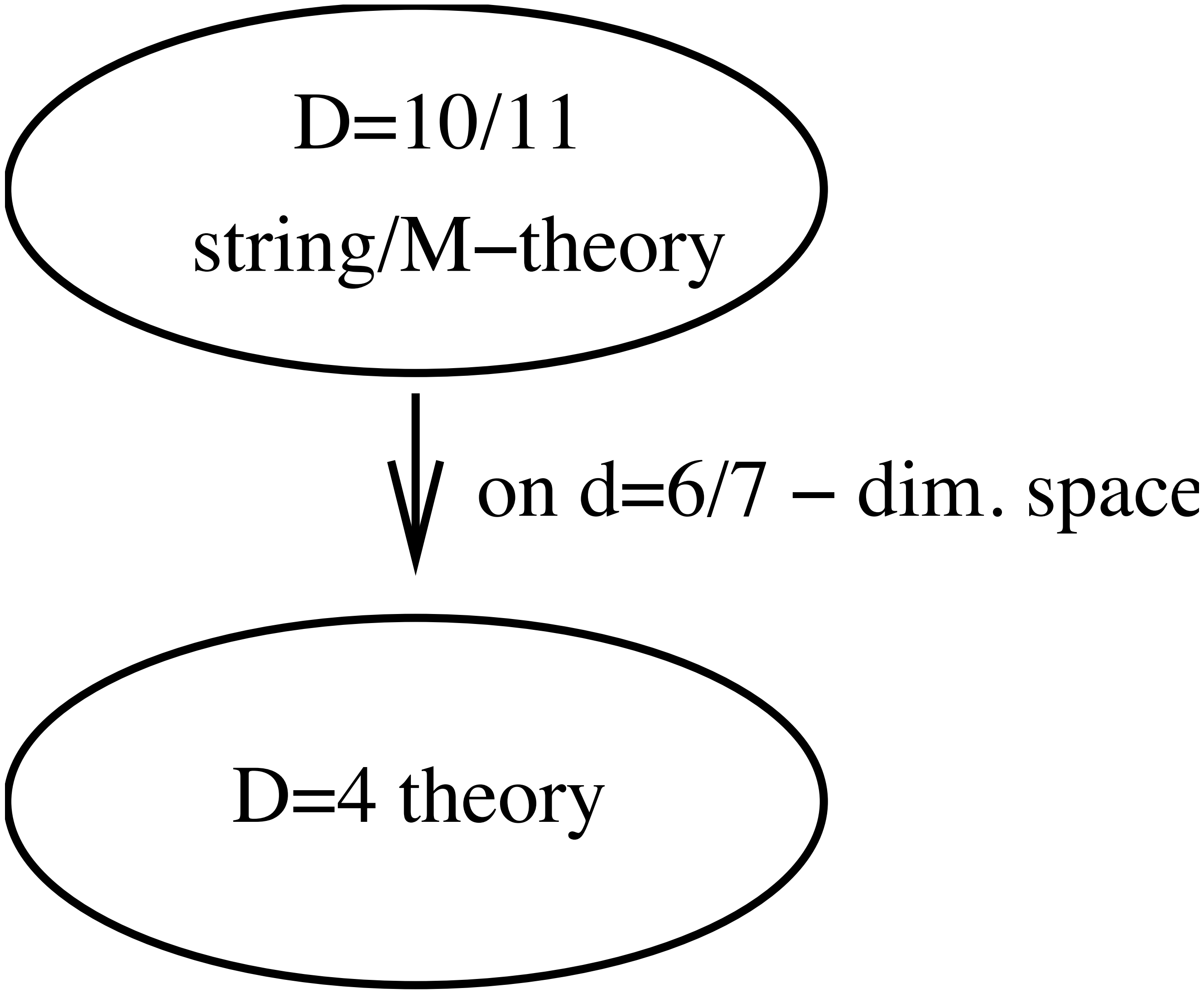
- spectrum: $\alpha' m^2 = n \in \mathbb{N}$ $\begin{cases} n = 0 & \rightarrow \quad \text{observed particles} \\ n > 0 & \rightarrow \quad \text{superheavy} \end{cases}$

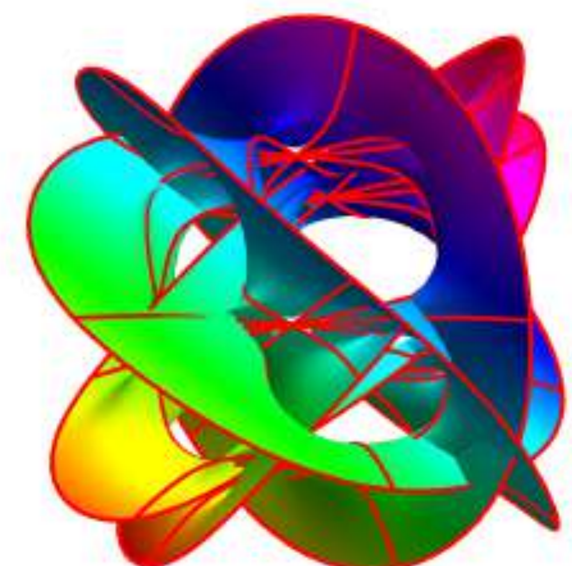
massless $(n = 0)$ modes contain: graviton (closed string)
gauge fields (open string)

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

Dimensions

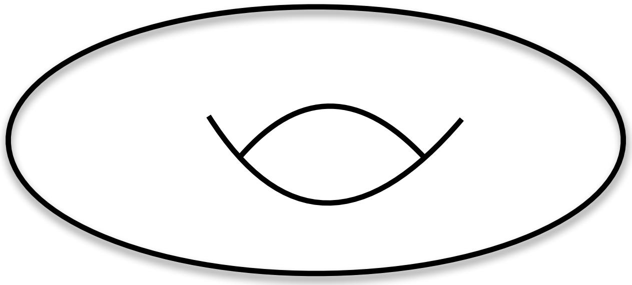
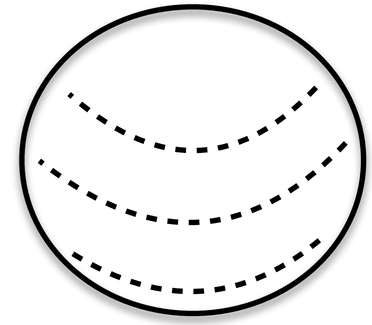
We need to "curl up" six (or seven) of the dimensions to make contact with physics -> compactification





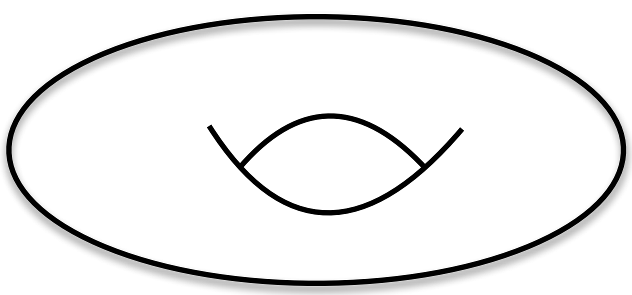
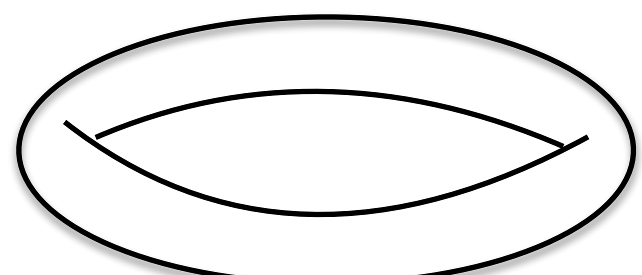
Calabi-Yau manifold
(bi-cubic)

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  ?

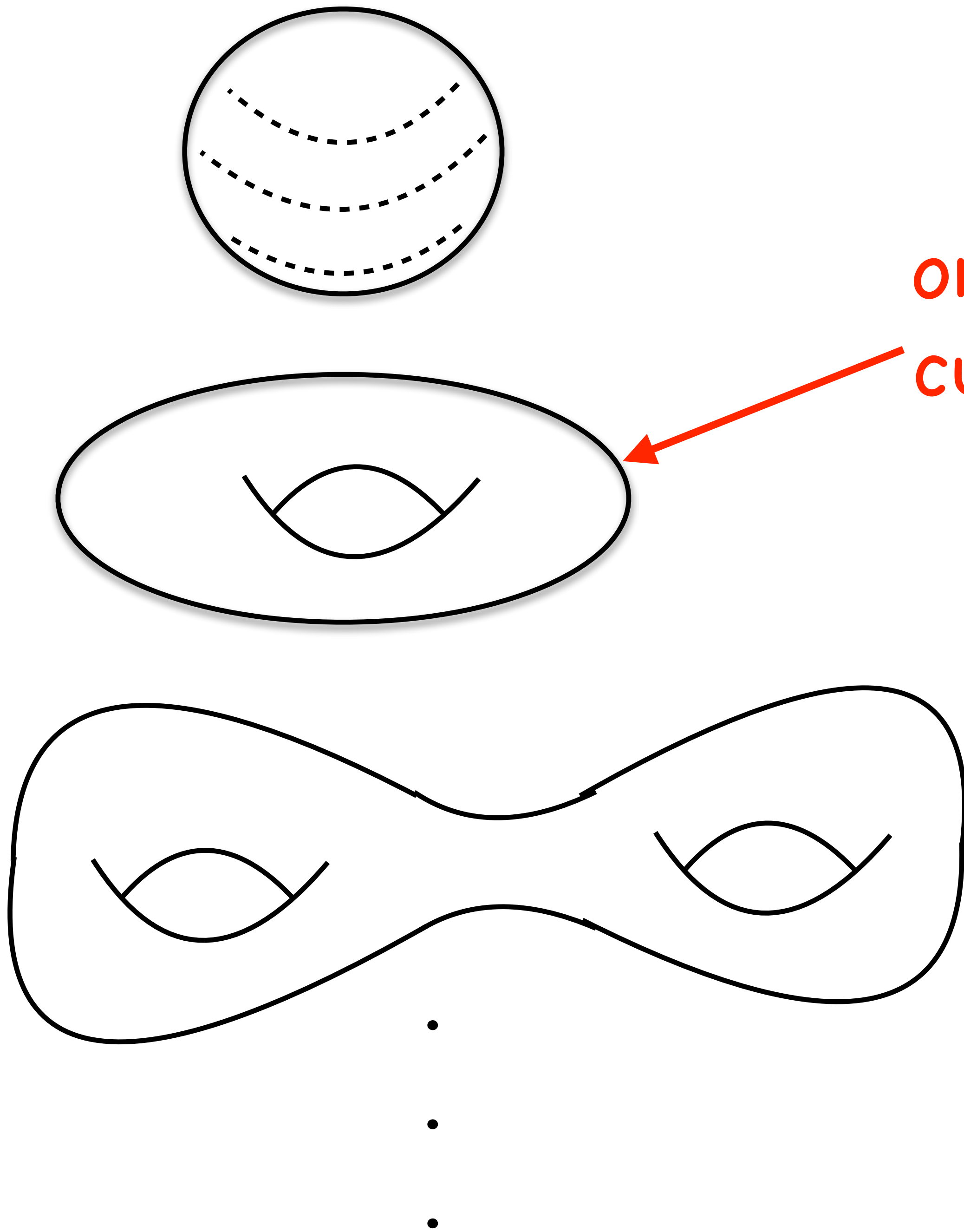
-> determines couplings/particle masses in 4d theory
(Maths: Differential Geometry)

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".

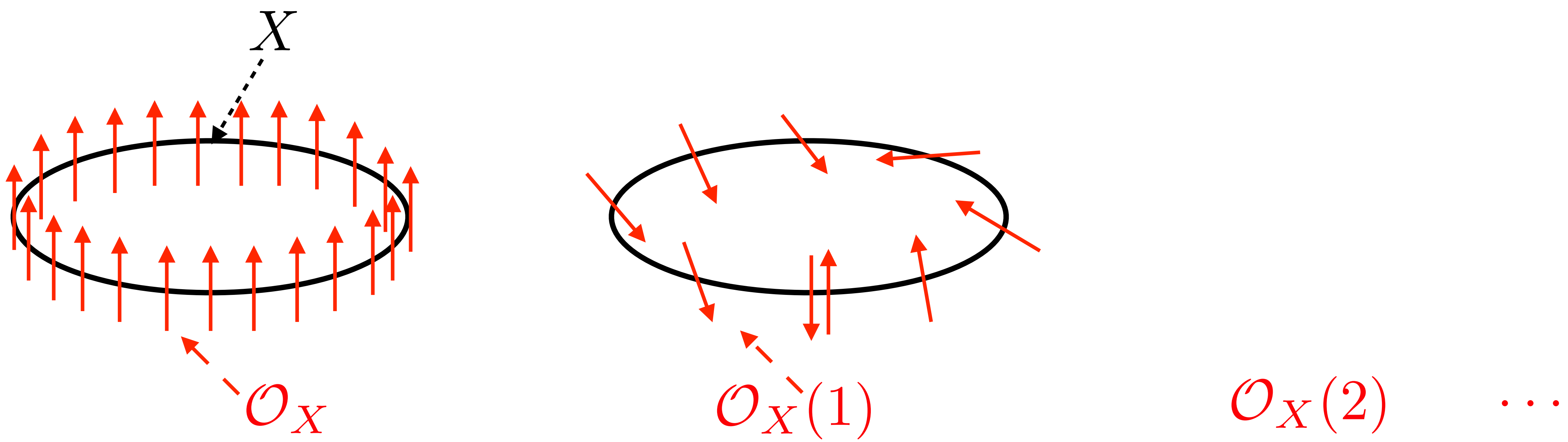
More generally, in 6d, it is classified by integer data -> many choices.

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?

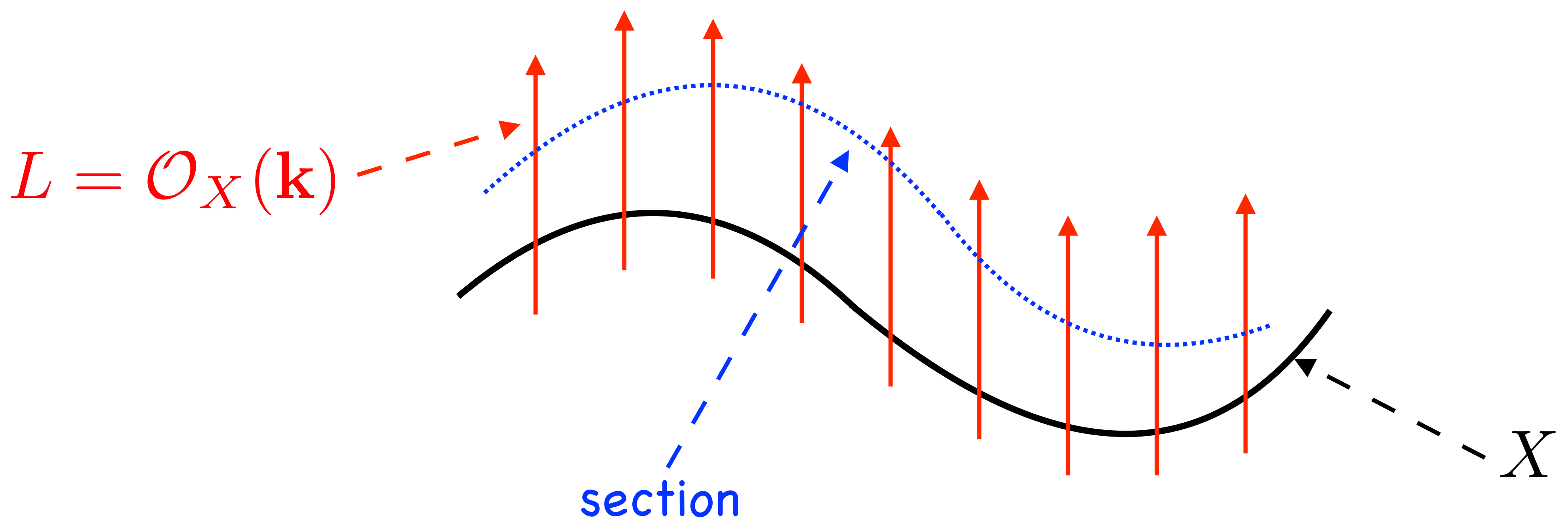
The space X carries additional structure, for example line bundles.



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

-> Example

Machine learning string theory

Learning line bundle cohomology

Q1: Can a machine learn the map $\mathbf{k} \to h^q(\mathcal{O}_X(\mathbf{k}))$?

Q2: Can a machine provide information about the mathematical structure of this map?

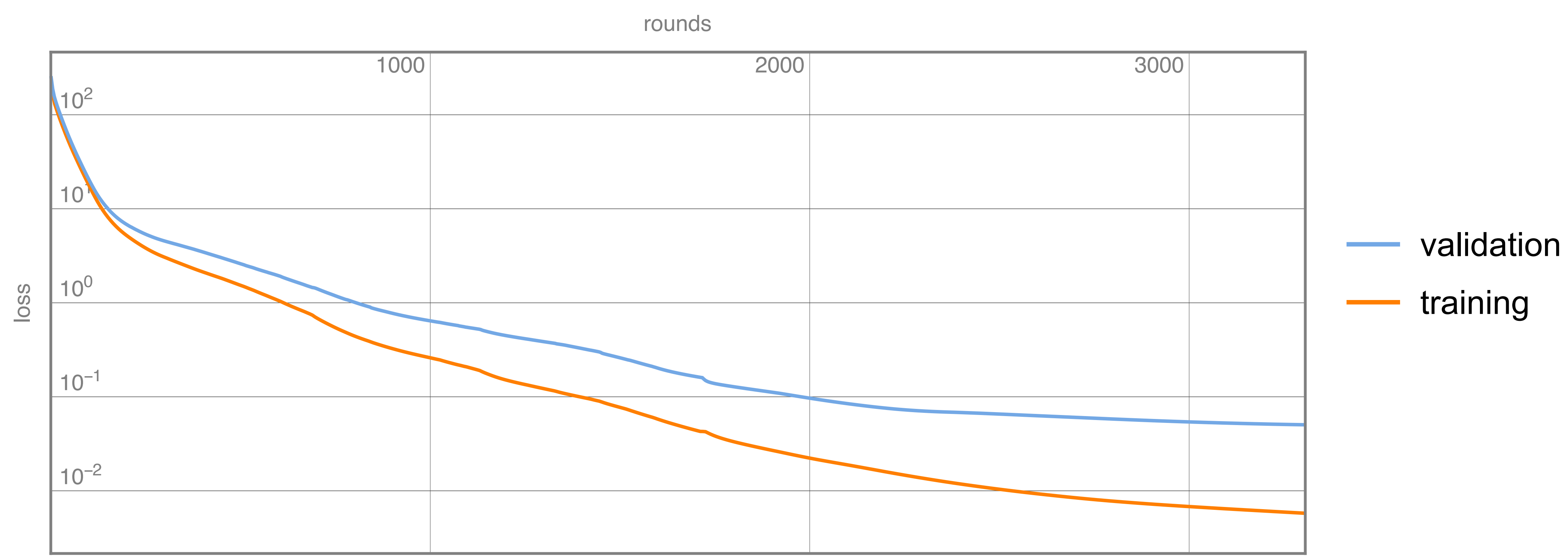
Training data: $\{\mathbf{k}_i, h^q(\mathcal{O}_X(\mathbf{k}_i)\}$ from horrendous calculations

Q1: Example for line bundle cohomology on $X = dP_2$

Want to learn $h^0(\mathcal{O}_X(\mathbf{k}))$ where $\mathbf{k} = (k_0, k_1, k_2)$.

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 spaces.

Advantages:

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

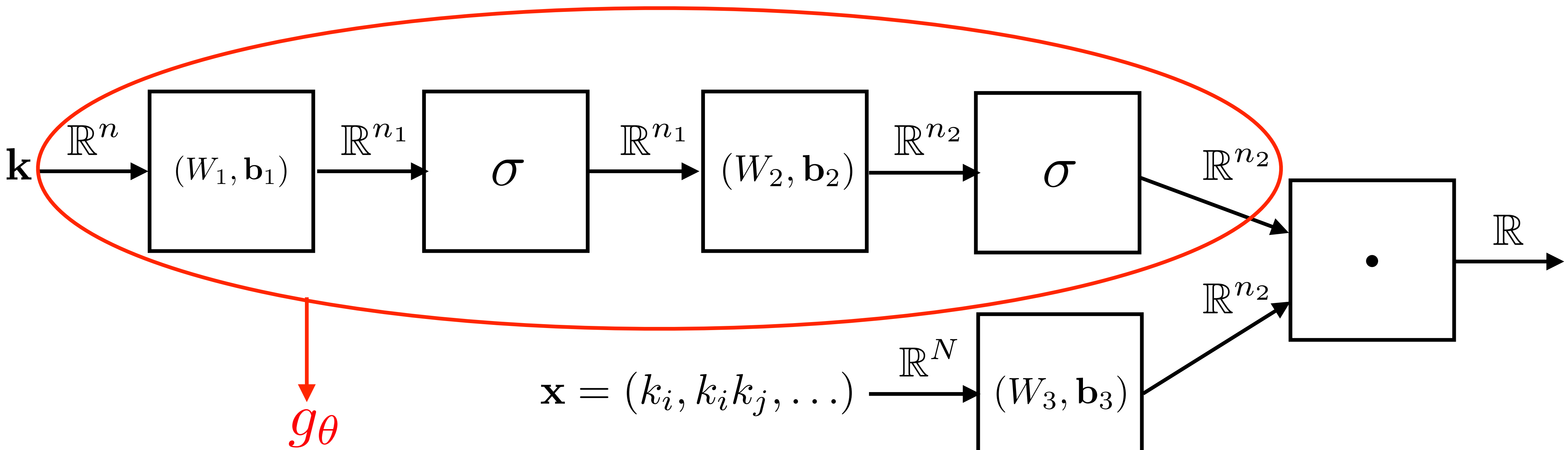
Disadvantages:

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

Q2: Can we use ML to conjecture formulae for $h^q(\mathcal{O}_X(\mathbf{k}))$?

Expectation: Formulae are "piecewise" polynomial

Design a net which matches the expected structure:



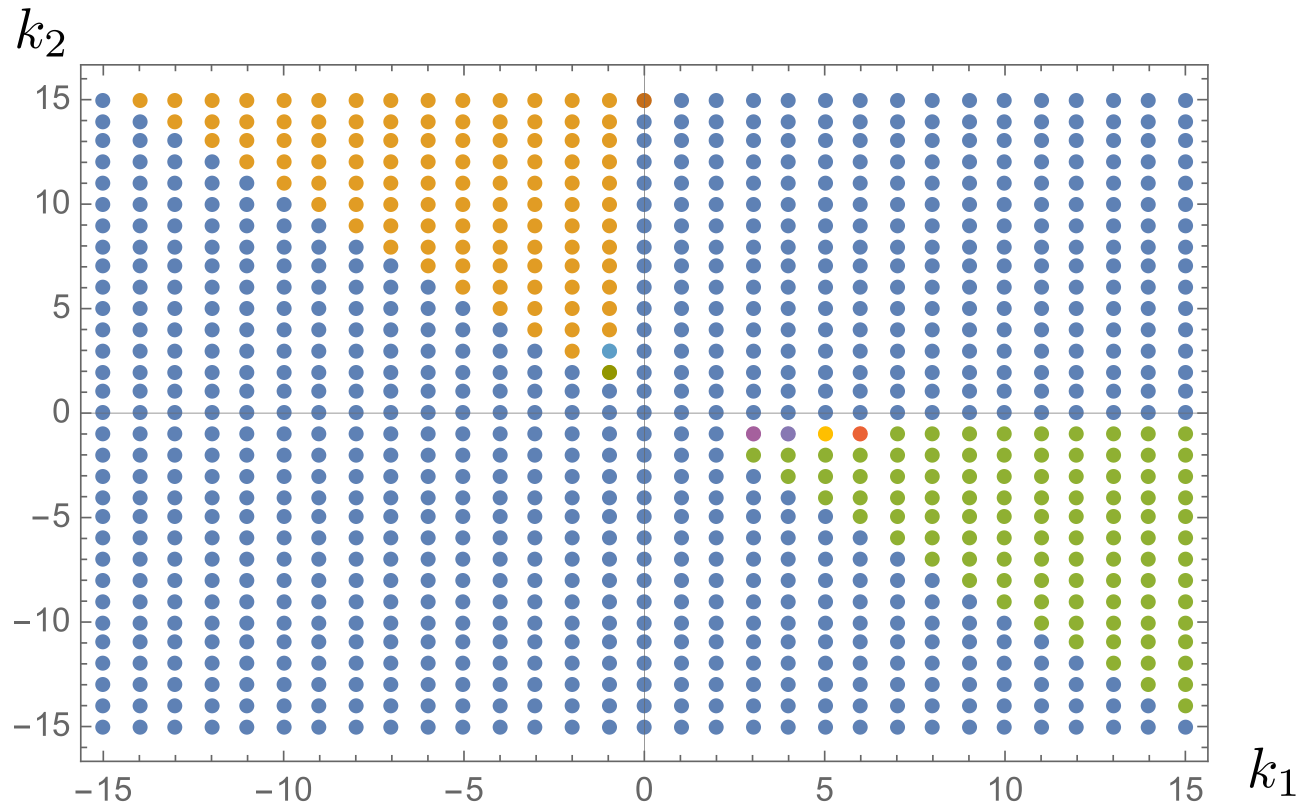
Assume net has been trained: $\rightarrow$ $g_{\theta_0},\ W_{30},\ \mathbf{b}_{30}$

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

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

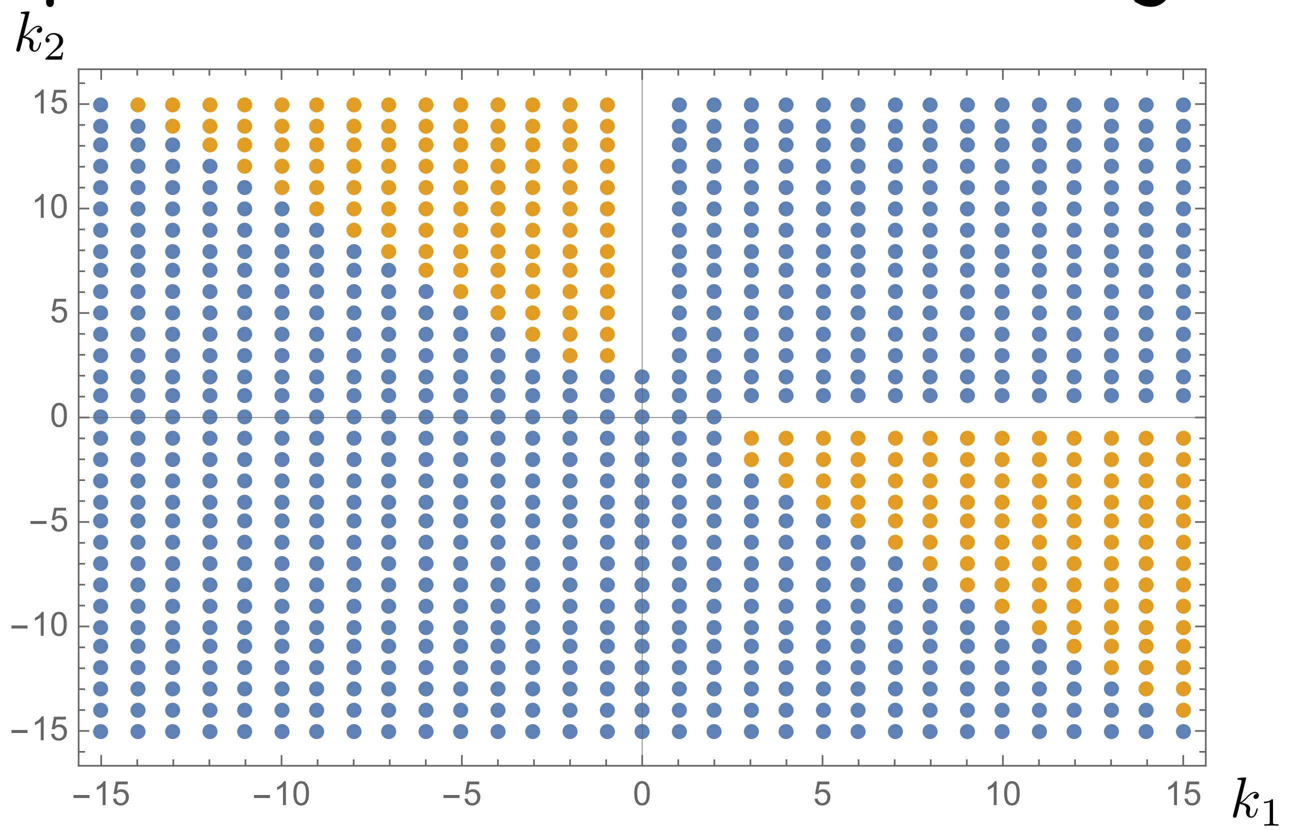


2) Find correct cubic polynomial for each region by a fit:

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:



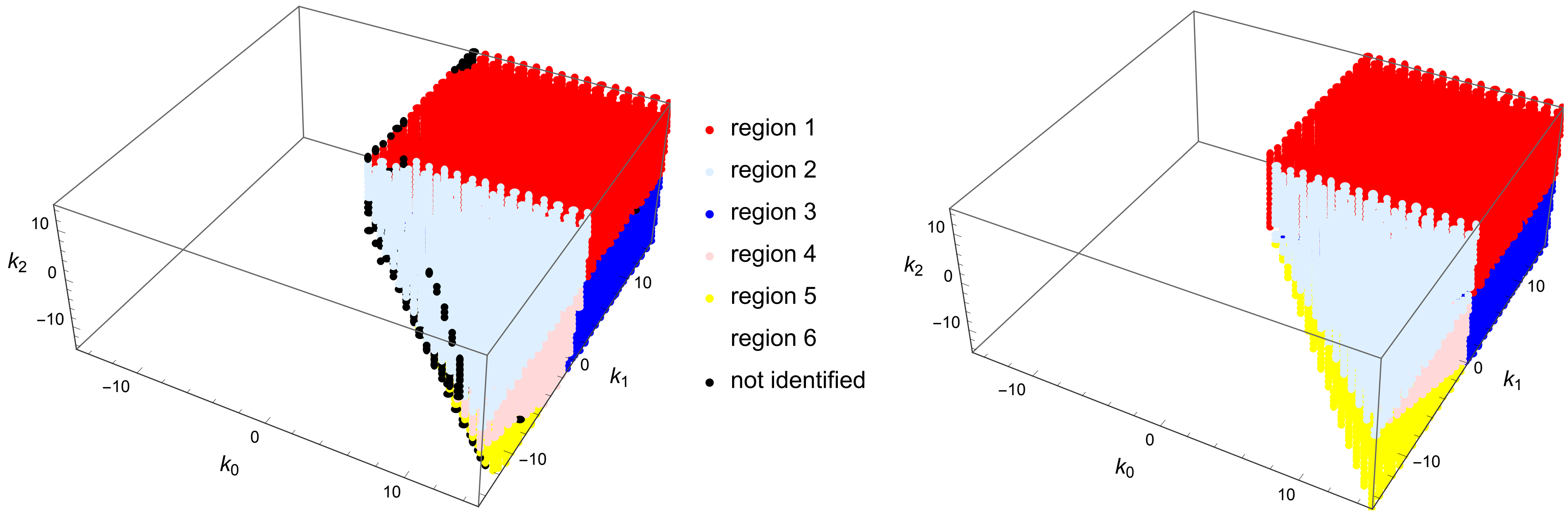
4) Find equations for boundaries of regions

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

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

Has been proved mathematically.

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



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

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					

Has also been proved mathematically.

ML can be used to generate mathematic conjectures.

Learning string theory standard models

A model with the right forces (strong, electro-weak) requires:

6d space X five line bundles $\mathcal{O}_X(\mathbf{k}_1), \ldots, \mathcal{O}_X(\mathbf{k}_5)$

Model characterised by integer matrix $K = (\mathbf{k}_1, \ldots, \mathbf{k}_5)$

For a given X, we can create a training set of the form

$$\{K \to 0 \text{ or } 1\}$$

non-SM SM

Q: Can ML distinguish standard models from non-standard models?

Example

$$X \in \left[\begin{array}{c|ccc} \mathbb{P}^1 & 0 & 1 & 1 \\ \mathbb{P}^1 & 0 & 1 & 1 \\ \mathbb{P}^1 & 1 & 1 & 0 \\ \mathbb{P}^1 & 1 & 1 & 0 \\ \mathbb{P}^1 & 1 & 0 & 1 \\ \mathbb{P}^1 & 1 & 0 & 1 \end{array}\right]$$

This space has ~ 17000 standard models, found by "brute force".
We also generate the same number of random non-SMs.

Two examples from the data set:

SM

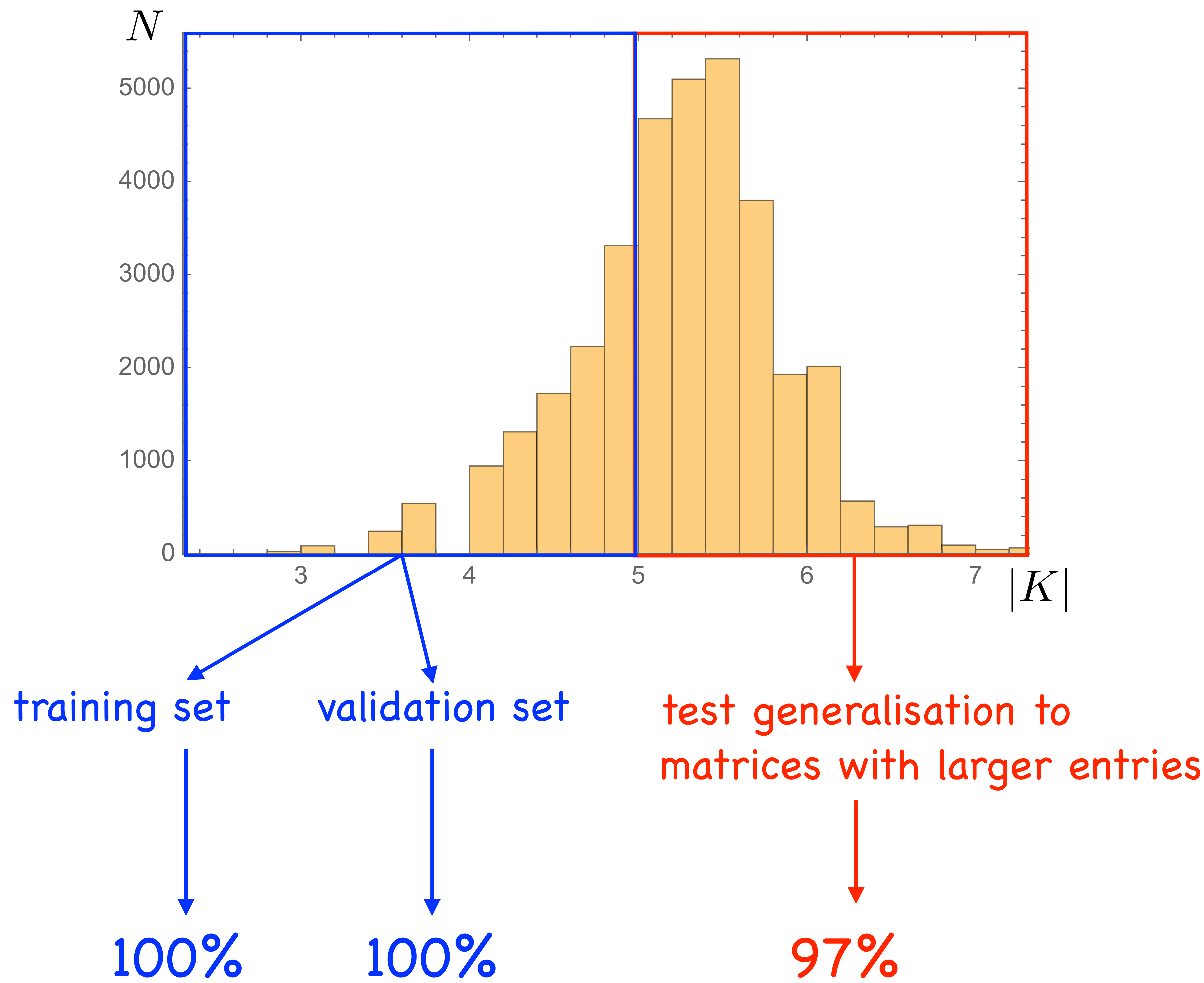
$$K = \begin{pmatrix} -1 & -1 & -1 & 1 & 2 \\ 0 & -2 & 0 & 1 & 1 \\ -1 & 1 & -1 & 0 & 1 \\ 1 & 0 & 1 & 0 & -2 \\ 0 & 1 & 0 & 0 & -1 \\ 1 & 0 & 1 & -2 & 0 \end{pmatrix} \to 1$$

non-SM

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

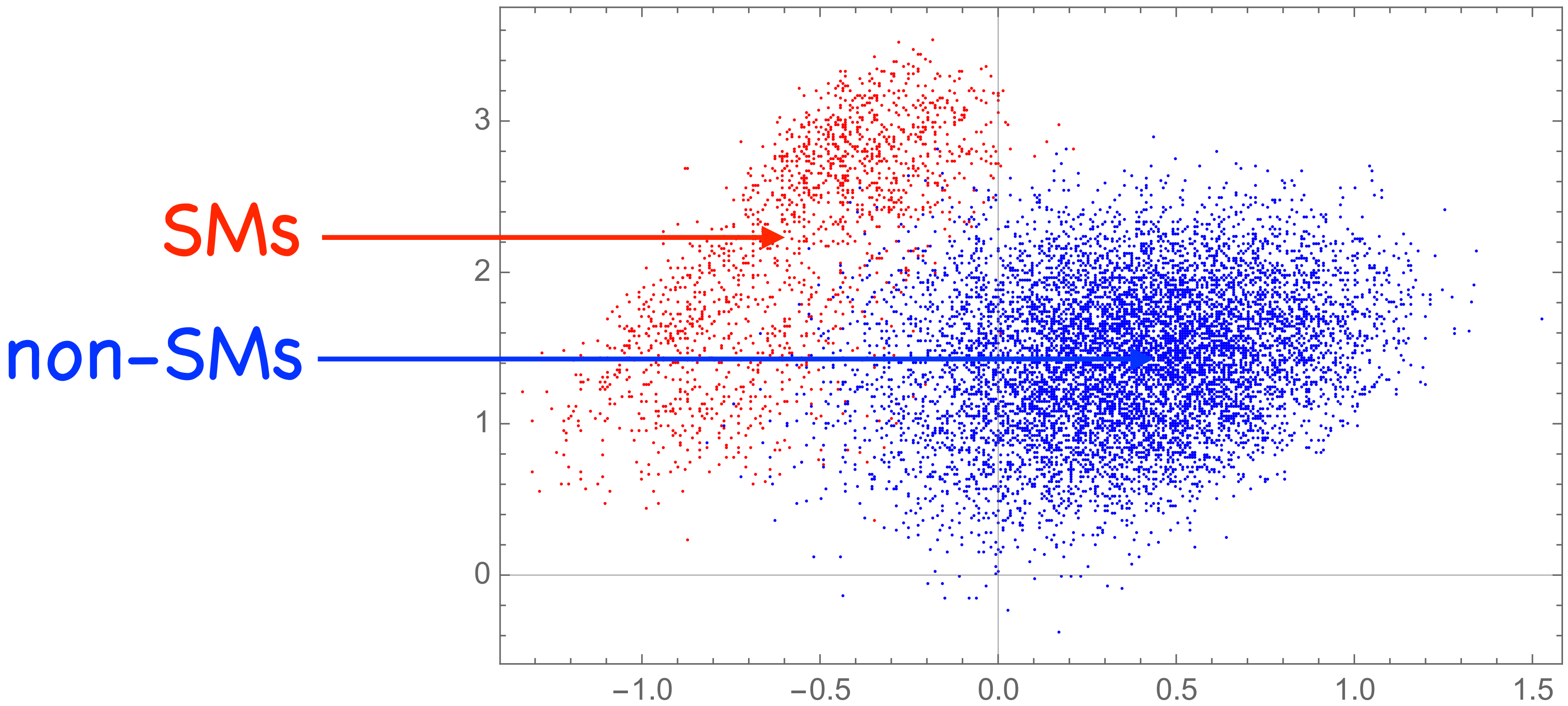
Can you tell them apart? Can the machine?

Network: simple 2 or 3 layer

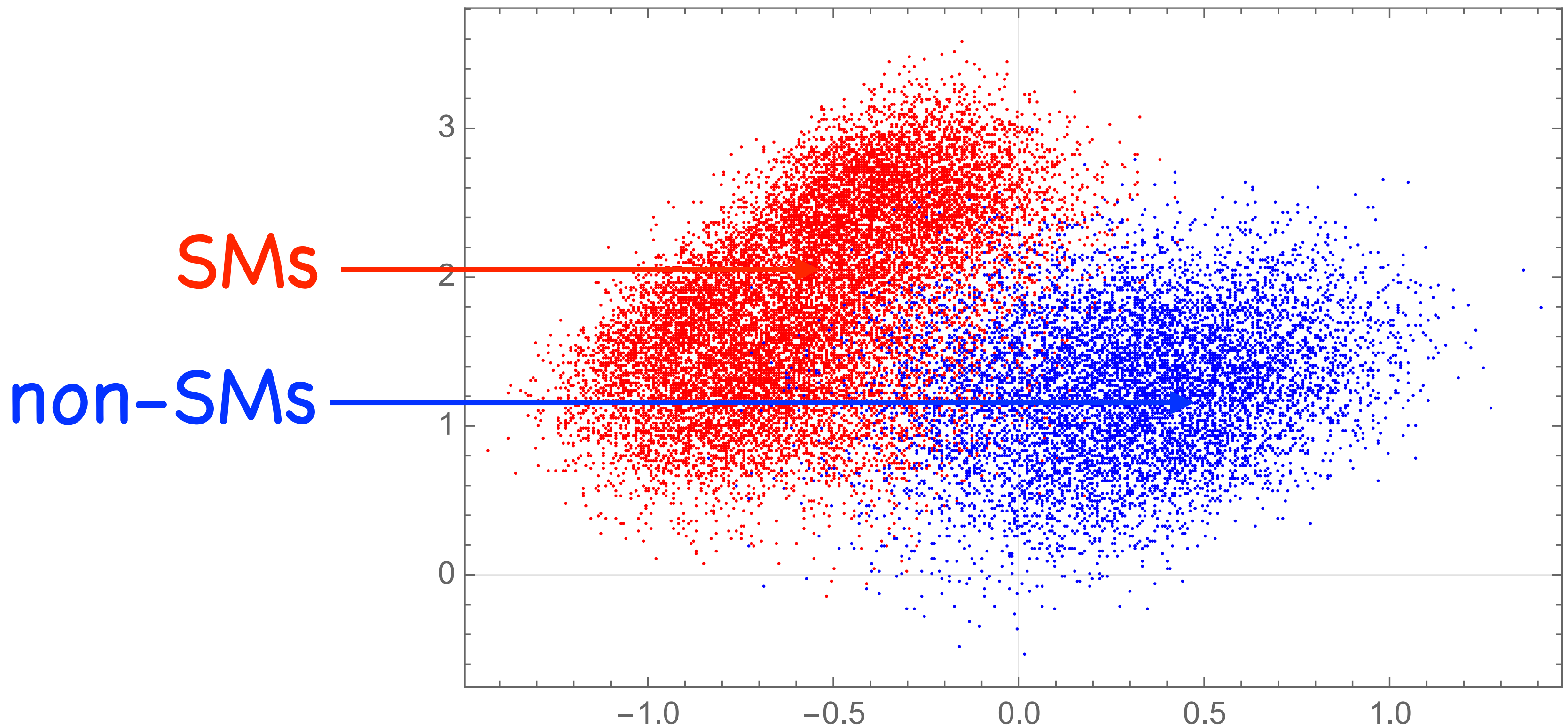


This provides a fast method to distinguish SM and non-SMs which works beyond the training range.
Still requires testing every matrix -> limited improvement.

Use auto-encoder on same data. Encoder maps into 2d space:



$|K| \leq 5$



$|K| > 5$

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

Conclusions

- ML in string theory is still in its infancy. The challenge is to match the right problems and techniques.

- ML can be used to generate non-trivial mathematical conjectures.

- ML can distinguish standard models from non standard models.

- Can ML techniques lead to substantial progress in string theory?

- Can the ``unusual" data sets and problems in string theory lead to insights into ML?

Thanks