Machine Learning in Particle and String Theory

Andre Lukas

University of Oxford

Colloquium, Institute for Basic Science, CTPU, Daejeon, South Korea, July, 2023

collaborators:
Steve Abel, Lara Anderson, Andreas Braun, Evgeny Buchbinder, Per Berglund, Callum Brodie, Andrei Constantin, James Gray, Thomas Harvey, Yang-Hui He, Elli Heyes, Edward Hirst, Vishnu Jejjala, Magdalena Larfors, Seung-Joo Lee, Challenger Mishra, Luca Nutricati, Burt Ovrut, Eran Palti, Fabian Ruehle, Robin Schneider, Sebastian von Hausegger

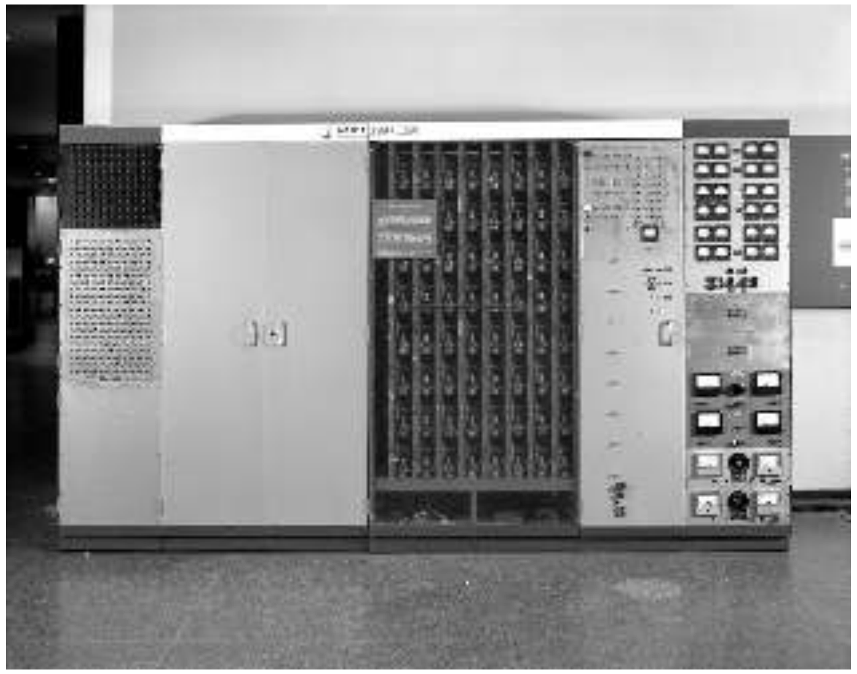
Outline

Introduction

A brief history of machine learning (ML)

Many underlying ideas for ML are known for a long time.

- A computational system simulating human brain function (Hebb 1949).
- First simple one-layer neural network (``perceptron”) (Rosenblatt 1957)

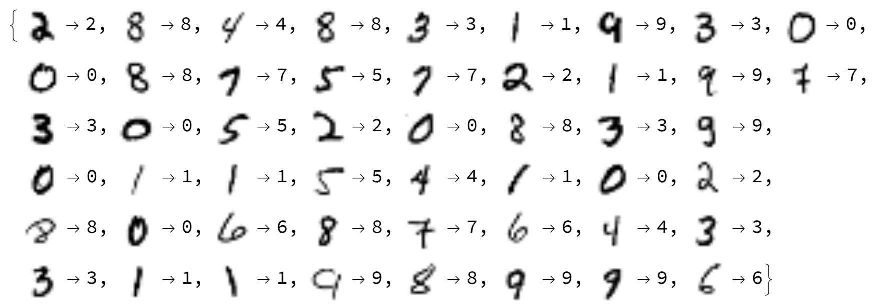


- Multi-layer (``deep”) neural networks (1960s)
- Training via back-propagation (=chain rule) (1970s)

But only more recently has computing power and available data become sufficient to uncover the potential of ML. Software realisations of NNs have lead to many new developments:

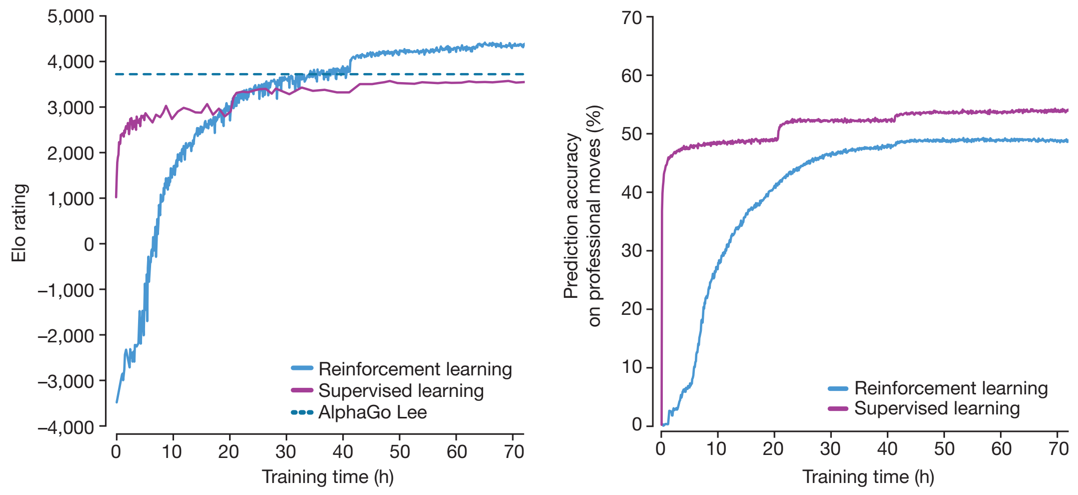
- Deep NNs with increasingly sophisticated architecture, for image/pattern/ speech recognition.

MNIST data set:



- Reinforcement learning (RL) to explore large environments (AlphaGo).

(Silver et al, DeepMind, Nature 2017)



- Transformer architecture underlying large language models (GPT3/4).

``a cross section of a walnut" ->

(Dall-E website)

ML will change our societies profoundly.

ML will have an enormous impact on how we pursue science.

ML is already used in many areas of physics, e.g.

- Event selection at the LHC.
- Image recognition in astronomy.
- Solving differential equations in Fluid Dynamics and General Relativity.
- Identifying promising physical models from data.

focus in this talk

Machine learning - a concise introduction

Neural networks - what are they?

A neural network (NN) is a family of functions

$$\boxed{\begin{array}{cccc} F_\theta : & \mathbb{R}^{d_{\text{in}}} & \rightarrow & \mathbb{R}^{d_{\text{out}}} \\ & \rotatebox{90}{$\in$} & & \rotatebox{90}{$\in$} \\ & x & \rightarrow & F_\theta(x) \end{array}}$$

parametrised by $\theta \in \mathbb{R}^n$.

As a diagram:

$$\mathbb{R}^{d_{\text{in}}} \ni x \longrightarrow \boxed{F_\theta} \longrightarrow F_\theta(x) \in \mathbb{R}^{d_{\text{out}}}$$

Building blocks of NNs

Usually, a neural network is formed from smaller building blocks

layers of NN $\longrightarrow$ $\boxed{f^{(i)}_{\theta_i}} : \mathbb{R}^{\boxed{d_{i-1}}} \to \mathbb{R}^{\boxed{d_i}}$, where $i = 1, \dots, \boxed{\delta}$,

width of NN

depth of NN
-> "deep learning"

by function composition, so

$$\boxed{F_\theta = f^{(\delta)}_{\theta_\delta} \circ \cdots \circ f^{(2)}_{\theta_2} \circ f^{(1)}_{\theta_1} : \mathbb{R}^{d_0} \to \mathbb{R}^{d_\delta}}$$

where $\theta = (\theta_1, \dots, \theta_\delta),\ d_{\text{in}} = d_0,\ d_{\text{out}} = d_\delta$.

As a diagram:

$$\begin{tikzcd}
\mathbb{R}^{d_{\text{in}}} \ni x \arrow[r] & \boxed{f^{(1)}_{\theta_1}} \arrow[r, "\mathbb{R}^{d_1}"] & \boxed{f^{(2)}_{\theta_2}} \arrow[r, "\mathbb{R}^{d_2}"] & \cdots \arrow[r, "\mathbb{R}^{d_{\delta-1}}"] & \boxed{f^{(\delta)}_{\theta_\delta}} \arrow[r] & F_\theta(x) \in \mathbb{R}^{d_{\text{out}}}
\end{tikzcd}$$

Example of building block - affine layer and activation

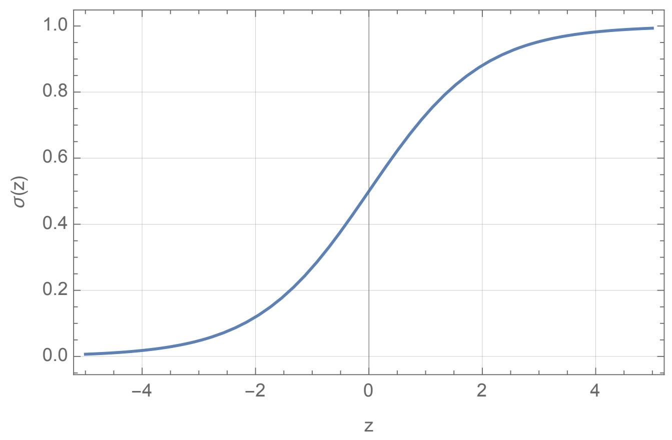
A common layer is an affine transformation followed by a function which acts in the same way on each vector component:

$$\boxed{\begin{aligned} f_{W,b} : \mathbb{R}^d &\to \mathbb{R}^{\tilde{d}} \qquad\qquad \theta = (W, b) \\ x &\mapsto f_{W,b}(x) = \sigma(Wx + b) \end{aligned}}$$

weights, $\tilde{d} \times d$ matrix

biases, $\in \mathbb{R}^{\tilde{d}}$

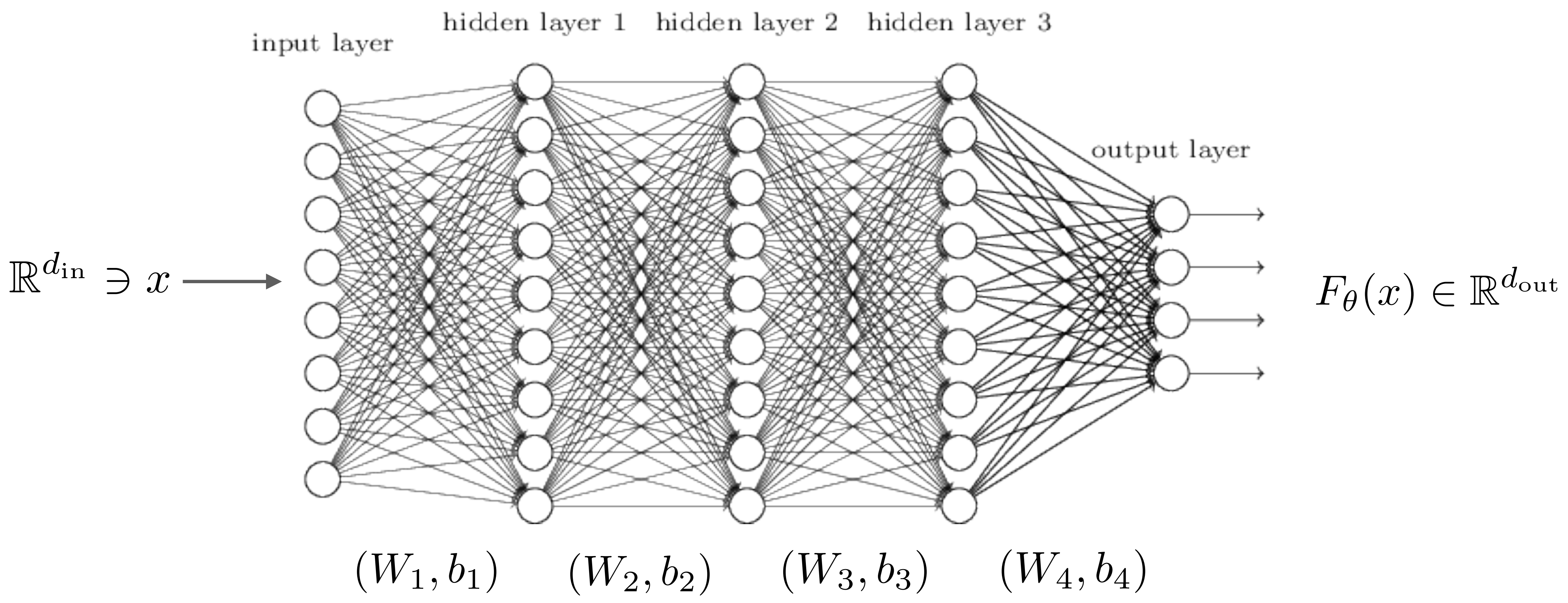
activation fct., e.g. logistic sigmoid: $\sigma(z) = \dfrac{1}{1 + e^{-z}}$



"fully connected network":

$$\boxed{F_\theta = f_{W_\delta, b_\delta} \circ \cdots \circ f_{W_2, b_2} \circ f_{W_1, b_1}}$$

A fully-connected network is often drawn as:



``A composition of affine transformations with mild non-linearities from the activation function". One layer -> perceptron.

Universal approximation theorem: For non-polynomial σ every continuous function can be approximated to arbitrary accuracy by a fully-connected NN, for sufficiently large width.

Example of building block - transformer block

A transformer block is a family of functions (From note by John Thickstun)

$$f_\theta : \mathbb{R}^{n\times d} \longrightarrow \mathbb{R}^{n\times d}$$

$$(\mathbf{x}_1, \ldots, \mathbf{x}_n) \mapsto (\mathbf{z}_1, \ldots, \mathbf{z}_n)$$

defined by

$$Q^{(h)}(\mathbf{x}_i) = W_{h,q}^T \mathbf{x}_i, \quad K^{(h)}(\mathbf{x}_i) = W_{h,k}^T \mathbf{x}_i, \quad V^{(h)}(\mathbf{x}_i) = W_{h,v}^T \mathbf{x}_i, \qquad W_{h,q}, W_{h,k}, W_{h,v} \in \mathbb{R}^{d\times k},$$

$$\alpha_{i,j}^{(h)} = \text{softmax}_j \left(\frac{\langle Q^{(h)}(\mathbf{x}_i), K^{(h)}(\mathbf{x}_j)\rangle}{\sqrt{k}} \right),$$

dot product

$$\mathbf{u}_i' = \sum_{h=1}^{H} W_{c,h}^T \sum_{j=1}^{n} \alpha_{i,j}^{(h)} V^{(h)}(\mathbf{x}_j), \qquad W_{c,h} \in \mathbb{R}^{k\times d},$$

"attention"

$$\mathbf{u}_i = \text{LayerNorm}(\mathbf{x}_i + \mathbf{u}_i'; \gamma_1, \beta_1), \qquad \gamma_1, \beta_1 \in \mathbb{R}^d,$$

$$\mathbf{z}_i' = W_2^T \text{ReLU}(W_1^T \mathbf{u}_i), \qquad W_1 \in \mathbb{R}^{d\times m}, W_2 \in \mathbb{R}^{m\times d},$$

$$\mathbf{z}_i = \text{LayerNorm}(\mathbf{u}_i + \mathbf{z}_i'; \gamma_2, \beta_2), \qquad \gamma_2, \beta_2 \in \mathbb{R}^d.$$

Large language models, such as GPT3/4, consist of a composition of transformer blocks.

What are neural networks for?

A: To approximate functions $g : \mathbb{R}^{d_{\rm in}} \to \mathbb{R}^{d_{\rm out}}$.

This is usually achieved by the following steps:

- Define a ``loss function" $L_g(\theta) \in \mathbb{R}$.
- Randomly initialise the parameters θ of the NN F_θ .
- Perform a gradient descent, i.e. iteratively change θ by

$$\boxed{\theta \mapsto \theta - \underset{\substack{\uparrow \\ \text{learning rate}}}{\lambda} \nabla_\theta L_g(\theta)}$$

``learning or training"

- The values θ_0 obtained in this way lead to an approximation for g , i.e.

$$\boxed{g \sim F_{\theta_0}}$$

Supervised learning

- The function g is specified in terms of data $(x_i, y_i) \in \mathbb{R}^{d_{\text{in}}} \times \mathbb{R}^{d_{\text{out}}}, \; i = 1, \ldots, N$.

  NN input, ``feature"

  desired NN output, ``label"

- Define a loss, e.g. mean square loss $L(\theta) = \frac{1}{N} \sum_i |F_\theta(x_i) - y_i|^2$

- Perform a (stochastic) gradient descent -> θ_0

- Test NN F_{θ_0} on unseen data.

-> examples

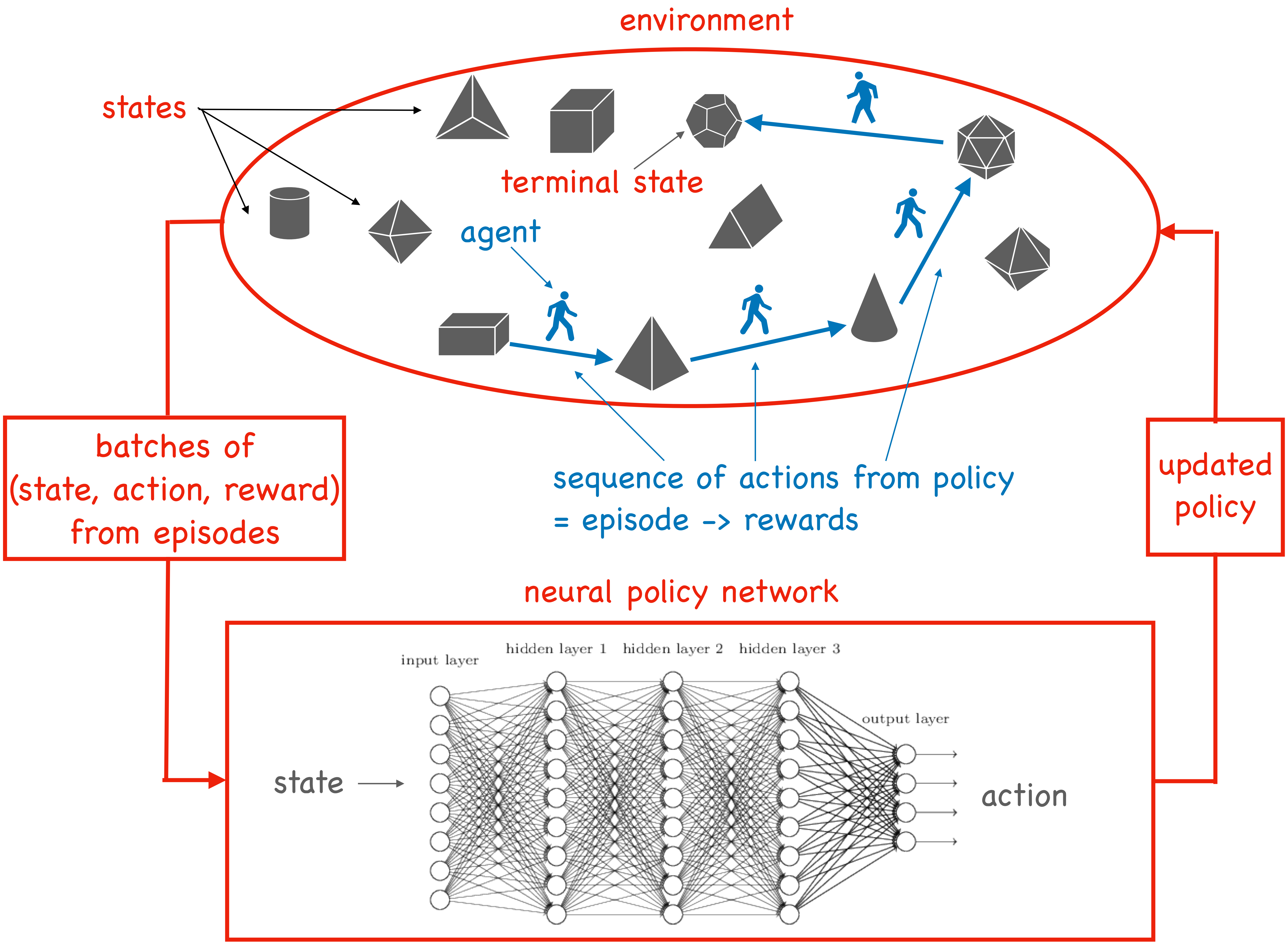
Self-supervised learning

- We have no labels but only features $(x_i) \in \mathbb{R}^{d_{\text{in}}}$, $i = 1, \ldots, N$.

- In addition, we know the function g must have a certain property $P(g) = 0$.

- Define a loss function $L(\theta) = \frac{1}{N} \sum_i |P(g(x_i))|^2$, otherwise proceed as before.

Reinforcement learning (RL)



Genetic algorithms

Basic ingredients of GAs

$$\begin{tikzcd}
 & \text{probability distribution} \arrow[d, dashed] & \text{genotypes} \arrow[dd, dashed] & \text{genotype/ phenotype map} \arrow[d, dashed] & \text{environment= phenotypes} \arrow[dd, dashed] & \text{fitness function} \arrow[d, dashed] & \\
 & p_{\text{in}} & & \varphi & & f & \\
{[0,1]} & & \mathbb{F}_2^{n_{\text{bits}}} \arrow[ll] \arrow[rr] & & E \arrow[rr] & & \mathbb{R} \\
 & & \rotatebox{90}{$\in$} & & \rotatebox{90}{$\in$} & & \\
 & & (1,0,0,1,\cdots,0,1) & & \text{physics models} & \text{``quality'' of model} \arrow[uu, dashed] &
\end{tikzcd}$$

Genetic evolution

(1) Sample with p_{in} to get initial population $P_0 \subset \mathbb{F}_2^{n_{\text{bits}}}$ with size $n_{\text{pop}} = |P_0|$.

(2) Evolve P_0 by combining (i) selection, (ii) cross-over and (iii) mutation:

$$P_0 \longrightarrow P_1 \longrightarrow P_2 \longrightarrow \cdots \longrightarrow P_{n_{\text{gen}}}$$

(3) Select all $b \in \bigcup_i P_i$ with $f \circ \varphi(b) \geq f_{\text{term}}$.
These are the terminal states b which lead to ``good models'' $\varphi(b)$.

Evolving $P_k \to P_{k+1}$

(i) selection:

- define probability distribution $p_k : \mathbb{F}_2^{n_{\text{bits}}} \to [0,1]$ by (roulette selection)

$$p_k(b) = \frac{1}{n_{\text{pop}}} \frac{(\alpha - 1)(f(\varphi(b)) - \bar{f}) + f_{\max} - \bar{f}}{f_{\max} - \bar{f}} \qquad \alpha \in [2,5]$$

- based on p_k, select $n_{\text{pop}}/2$ pairs of individuals from P_k

(ii) cross-over:

- for each pair from (i), pick a random position $k \in \{1, \dots, n_{\text{bits}}\}$
- swap tails of two individuals:

$$\begin{aligned} b_1 &= (1,0,\dots,1,0,\overset{k^{\text{th}}}{\boxed{1,1,1,0,\dots,1,1,0,0}}) & & \tilde{b}_1 = (1,0,\dots,1,0,\boxed{0,0,1,0,\dots,0,1,0,1}) \\ & \qquad\qquad\qquad \updownarrow & \longrightarrow & \\ b_2 &= (0,0,\dots,0,1,\boxed{0,0,1,0,\dots,0,1,0,1}) & & \tilde{b}_2 = (0,0,\dots,0,1,\boxed{1,1,1,0,\dots,1,1,0,0}) \end{aligned}$$

- the n_{pop} new individuals obtained in the way form a population $\tilde{P}_{k+1}$

(iii) mutation:

- randomly flip a small fraction $r \sim 0.01$ of bits in $\tilde{P}_{k+1}$ to obtain P_{k+1}

Differential equations

Basic idea

Suppose we have a differential equation for a function g of the form

$$p(D^2 g(x), Dg(x), g(x), x) = 0$$

We would like to solve this equation with self-supervised ML.

- Select a training set of points (x_i), $i = 1, \ldots, N$.
- Select a NN F_θ - this represents the prospective solutions g
- Define a loss function $L(\theta) = \frac{1}{N} \sum_i |p(\underbrace{D^2 F_\theta(x_i), DF_\theta(x_i)}_{\text{efficient tools to differentiate NNs exist}}, F_\theta(x_i), x_i)|^2 + \cdots$
- Perform a gradient descent with this loss function -> $g \sim F_{\theta_0}$.

How does this compare to lattice methods?

Suppose we are working in d (space-time) dimensions, so $x \in \mathbb{R}^d$

lattice	NN
``curse of dimension'' $N \sim (\#\text{points}/\text{dim})^{\boxed{d}}$	$N = ?$
solve N coupled eqs.	gradient of L for each batch
solution on lattice pts. x_i	NN F_{θ_0} is solution, everywhere

What does this mean in practice?

Example: Metric on Calabi-Yau (CY) manifolds in string theory

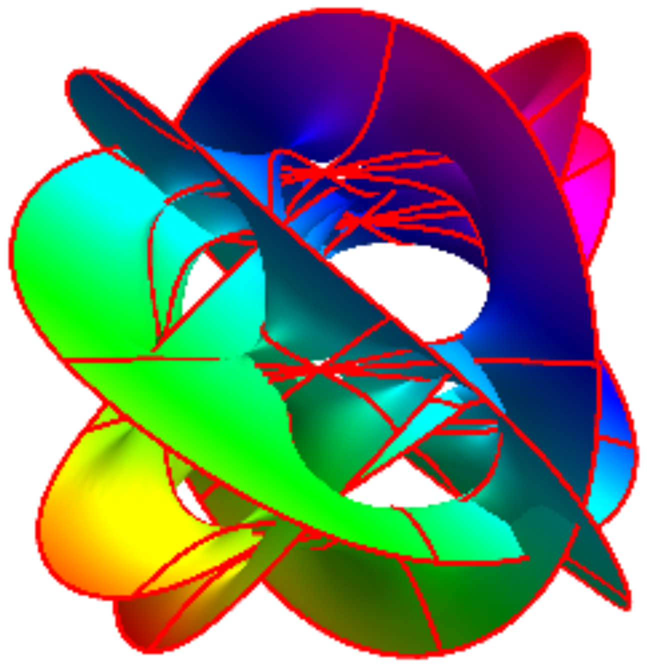
Why string theory and why CY manifolds?

- String theory is a consistent theory which contains gauge theories and (quantum) gravity.
- But it is defined in 10 space-time dimensions.
- To make contact with physics we need to compactify (``curl up”) 6 dimensions.

$$\begin{tikzcd}
\text{10d string theory} \arrow[d, "\begin{array}{l}\text{compactify on}\\ \text{6d manifold } X\end{array}"] \\
\text{4d QFT}
\end{tikzcd}$$

But we need to satisfy the (10d) Einstein equations, so X needs to carry a metric with vanishing Ricci tensor.

Yau's theorem: ``Ricci-flat metrics exist (and are unique under certain extra conditions) on CY manifolds."



= bi-cubic

The 10d theory is (basically) unique, but the 4d theory depends on X.

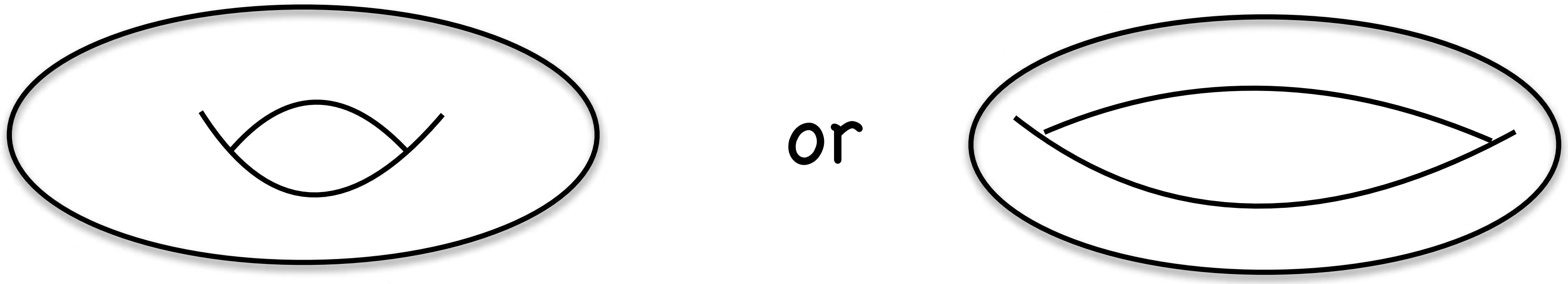
How does the 4d theory depend on X?

topology : ?

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

Many compactification of string theory known which lead to the forces and particle content of the standard model (SM) of particle physics!

main focus so far -> example later

shape :  ?

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

Can string theory also explain the couplings and masses in the SM?
``Can string theory explain the electron mass?"

to tackle this we need the Ricci-flat metric g on X <-> shape

Ricci-flat CY metrics from ML

(M. Larfors, AL, F. Ruehle, R. Schneider, 2211.010436, 2205.13408)

Not a single Ricci-flat CY metric known analytically -> numerical methods

First consider lattice methods: $(20 \text{ points/dim})^6 = 6.4 \times 10^7$ points

ML approach:

- Generate point sample (x_i), $i = 1, \ldots, N$, on CY X
- Use fully-connected NN F_θ
- Loss function $L(\theta) = \dfrac{1}{N} \sum_i |\mathrm{Ricci}(g(x_i))|^2 + \cdots$
- Perform gradient descent

Training

!!!!

(3 hidden layer, width 64, GELU activation, 100000 points each, Adam optimiser)

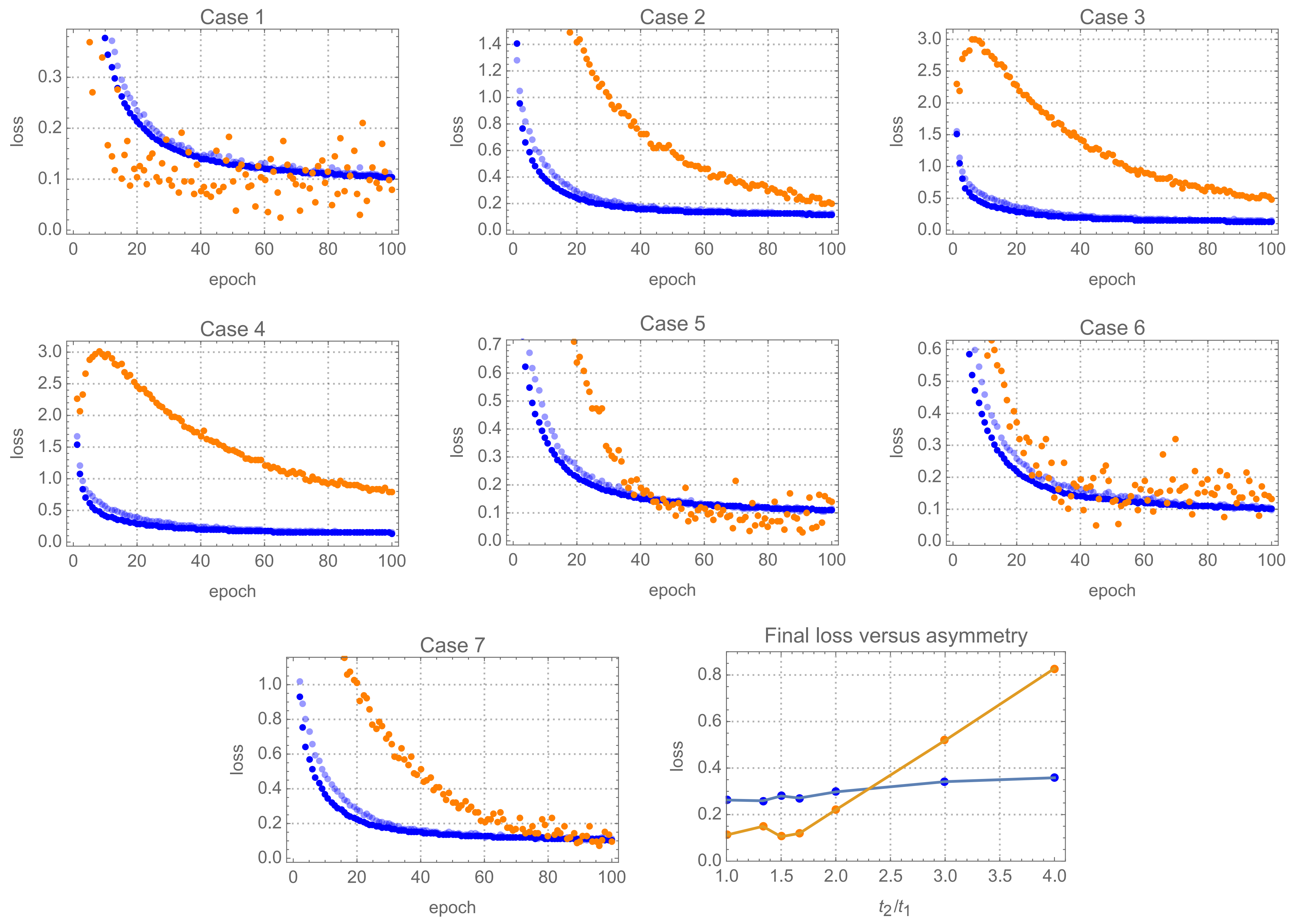


Figure 2: Bi-cubic training curves for the seven choices of Kähler parameters in Table 2. The last plot represents the final loss, obtained by averaging over the last 10 epochs, as a function of t^2/t^1 (orange: $\mathcal{L}_{\mathrm{Kclass}}$, blue: $4 \times \mathcal{L}_{\mathrm{MA}}$, both on training data, light-blue: $4 \times \sigma$ measure on validation data).

Check: compute volume of CY manifold

$$V_{\text{int}} = \frac{1}{6} d_{\alpha\beta\gamma} t^\alpha t^\beta t^\gamma \ , \ V_{\text{FS}} = \frac{1}{N} \sum_{i=1}^{N} \tilde{w}_i \det(g_{\text{FS}}(p_i)) \ , \ V_{\text{CY}} = \frac{1}{N} \sum_{i=1}^{N} \tilde{w}_i \det(g_{\text{CY}}(p_i))$$

case	1	2	3	4	5	6	7
V_{int}	8.49	4.97	2.93	2.02	6.87	7.59	6.16
V_{FS}	8.49	4.50	2.94	2.03	6.91	7.58	6.26
error	$< 1\%$	$< 1\%$	$< 1\%$	$< 1\%$	$< 1\%$	$< 1\%$	$\sim 2\%$
V_{CY}	8.56	5.03	2.96	2.03	6.86	7.58	6.28
error	$< 1\%$	$\sim 1\%$	$< 1\%$	$< 1\%$	$< 1\%$	$< 1\%$	$\sim 2\%$

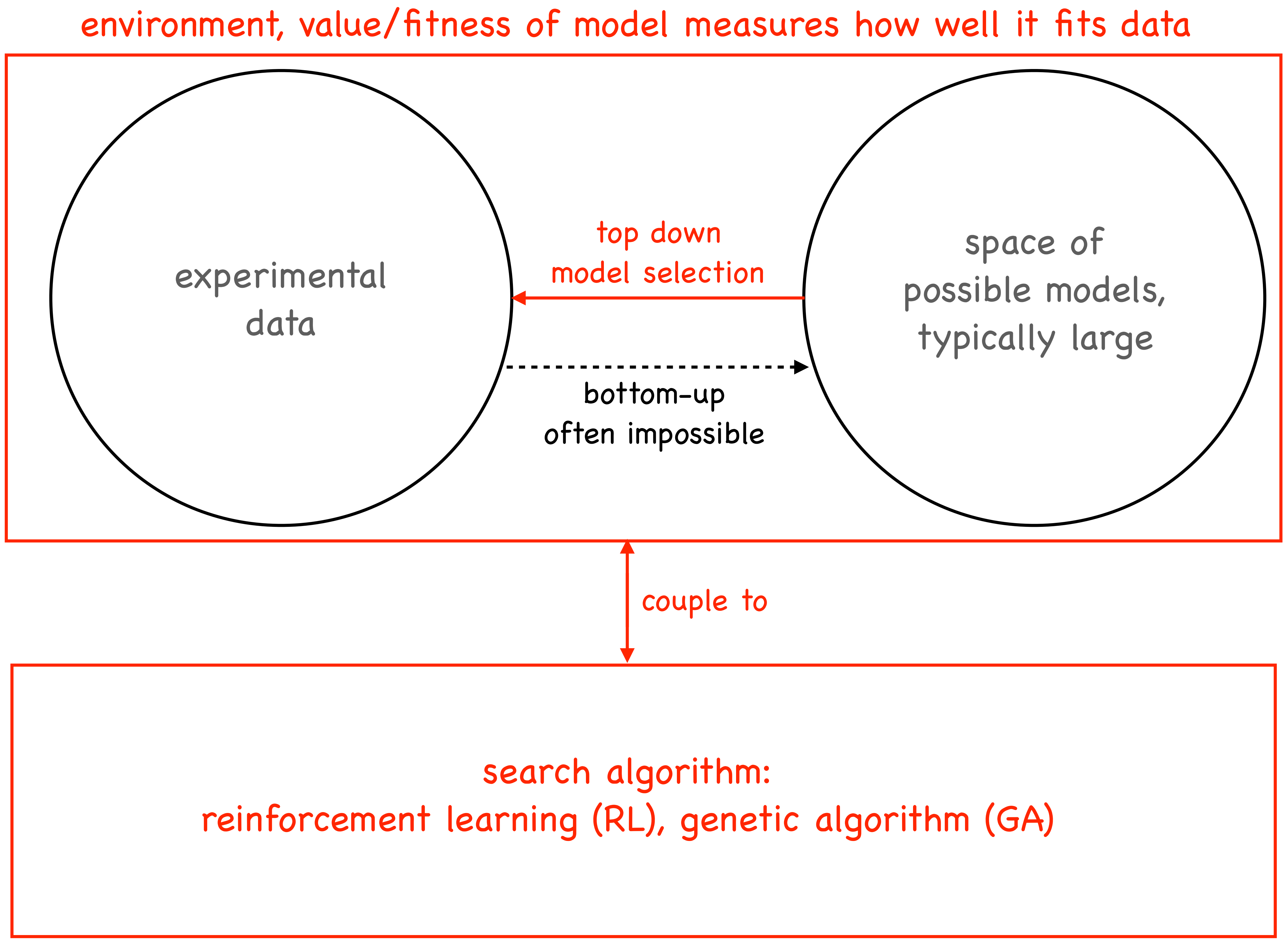
Table 3: Exact volume from intersection form (row 2), and volume from numerical integration with g_{FS} (row 3) and g_{CY} (row 4), for the seven choices of Kähler parameters in Table 2.

- accurate CY volume -> Ricci-flat metric accurate

This provides a crucial tools for computing particle masses from string theory!

Model building

Basic idea


environment, value/fitness of model measures how well it fits data
experimental data
top down model selection
bottom-up often impossible
space of possible models, typically large
couple to
search algorithm: reinforcement learning (RL), genetic algorithm (GA)

Example 1: heterotic CY models with flux (=bundles)

(A, Constantin, AL, T. Harvey, 2108.07316)

Consider bi-cubic CY X with flux = vector bundle V defined by

$$0 \to V \to B \xrightarrow{f} C \to 0 \qquad V \cong \mathrm{Ker}(f)$$

``monad"

$$B = \bigoplus_{a=1}^{r_B} \mathcal{O}_X(\mathbf{b}_a) \qquad C = \bigoplus_{\alpha=1}^{r_C} \mathcal{O}_X(\mathbf{c}_\alpha) \qquad r_B - r_C = 4$$

2d integer vectors

A model is described by a $2 \times (r_B + r_C)$ integer matrix

$$\{\ (\mathbf{b}_1, \ldots, \mathbf{b}_{r_B}, \mathbf{c}_1, \ldots, \mathbf{c}_{r_C})\ \} \ = \ \text{environment}$$

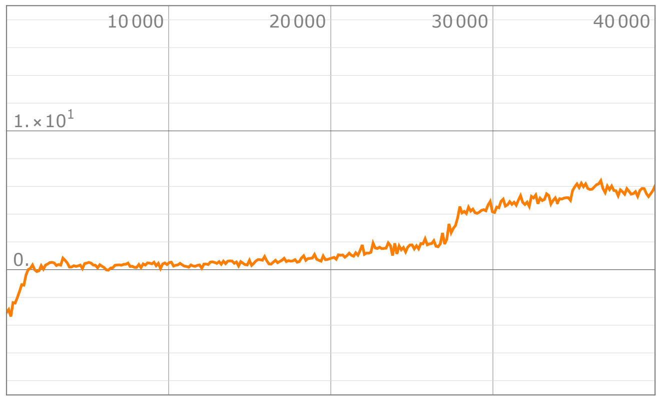
The particle content of a 4d model can be computed from this matrix.
(But it's complicated!)

Goal: Find models which lead to a SM spectrum.

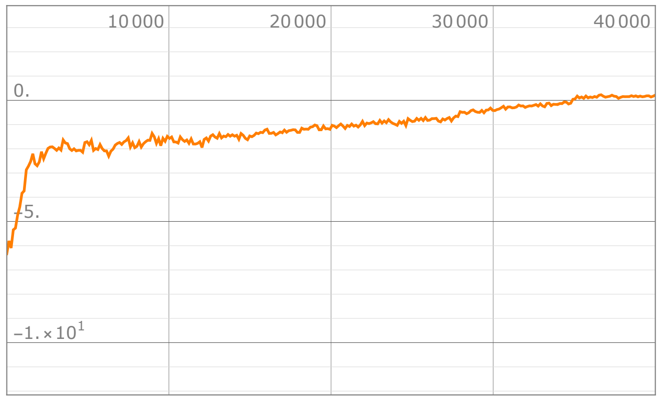
size of environment: $\sim 10^{2(r_B+r_C)} \overset{r_B=5,\ r_C=1}{\swarrow\!=} 10^{12}$

Example RL run for bi-cubic (actor-critic): $r_B = 6,\ r_C = 2 \quad \to \quad \#\text{states} \sim 10^{16}$

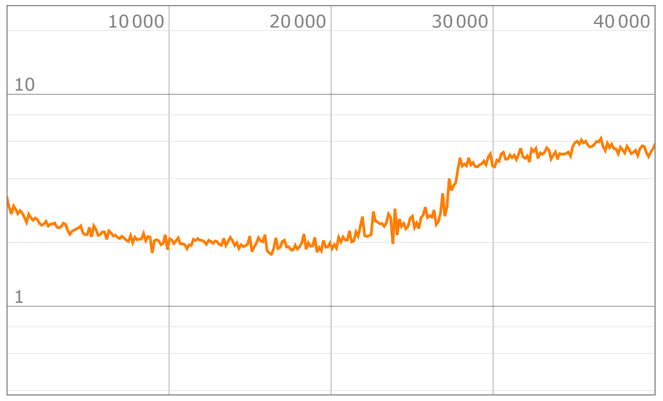
Training: about 1h on a single CPU



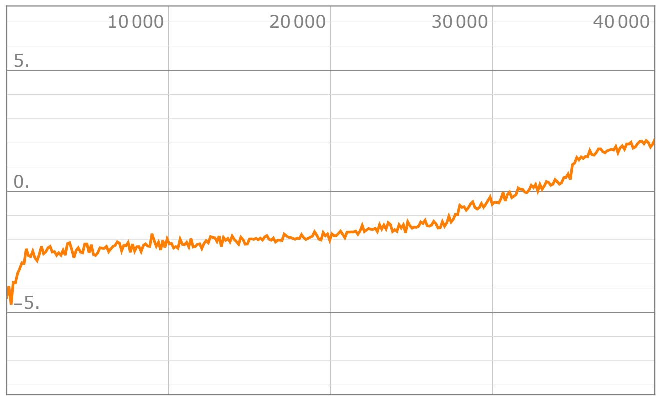
(a) Loss vs batch number.



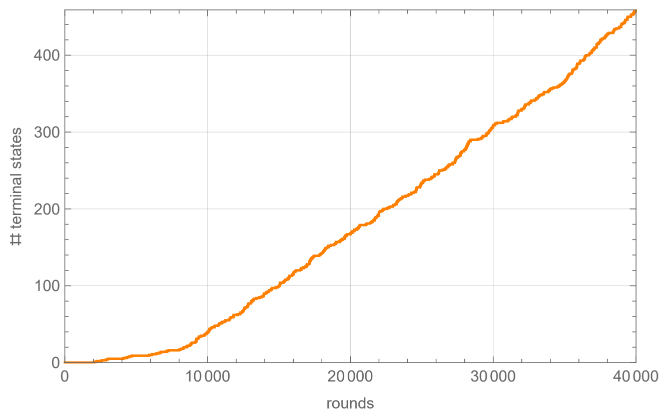
(b) Policy loss vs batch number.



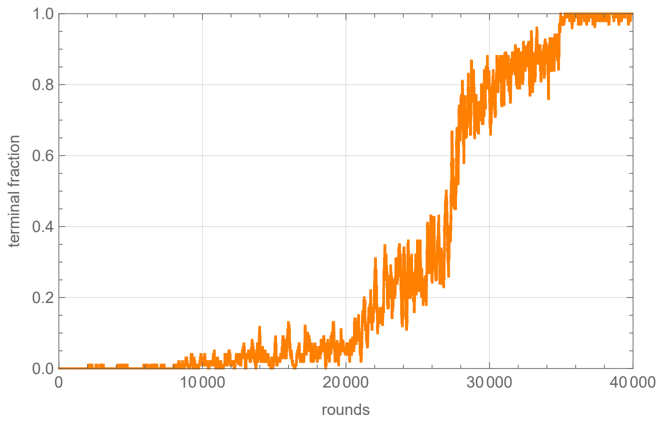
(c) Value loss vs batch number.



(d) TD return vs batch number.



(e) Number of terminal states vs episode number.



(f) Terminal fraction vs episode number.

Figure 6: Training metrics for the bicubic monad environment with $(r_B, r_C) = (6, 2)$.

Results: O(500) candidate models -> 18 new models with SM spectrum

Example 2: search for inflationary models

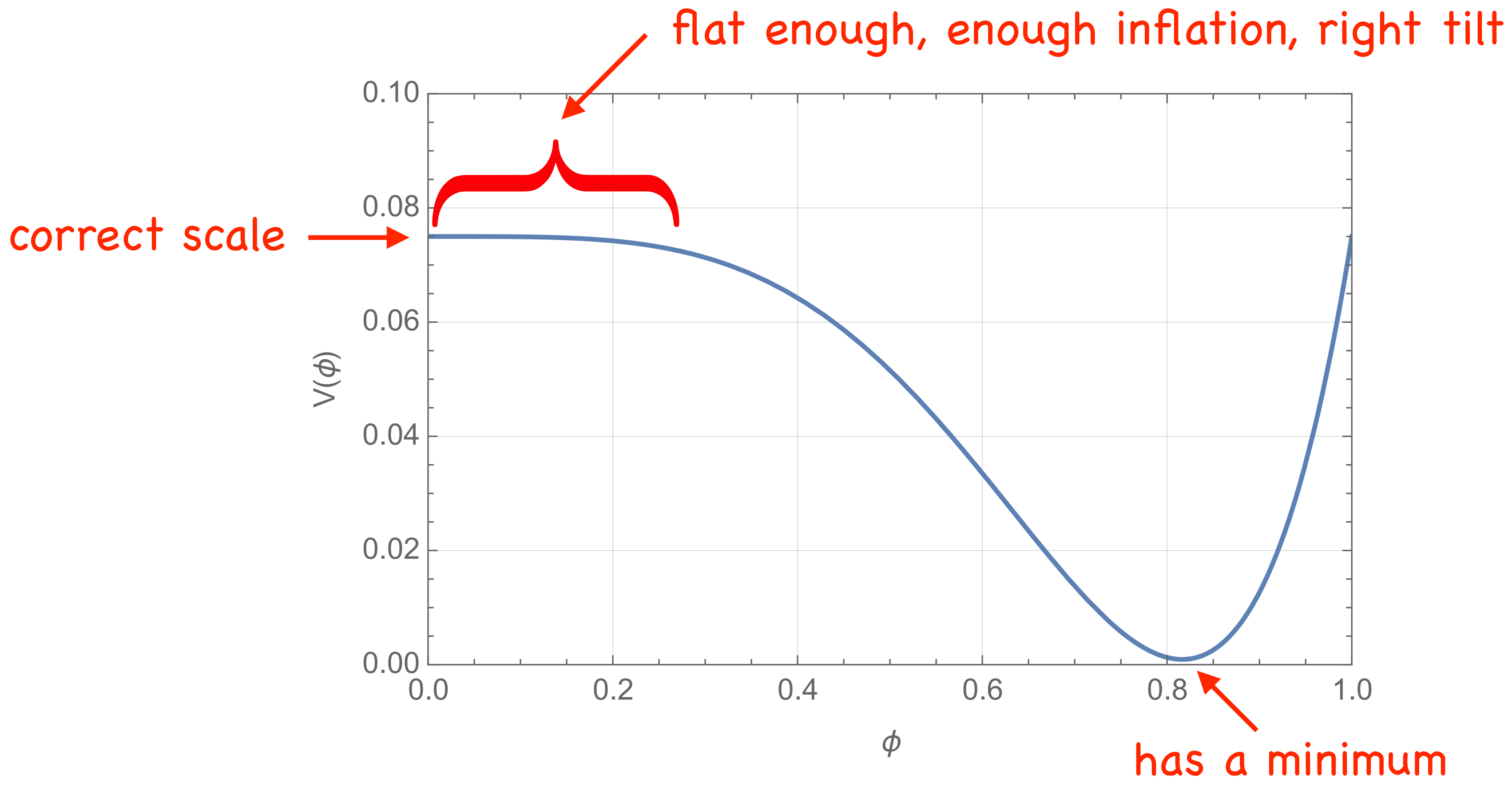
(S. Abel, A. Constantin, T. Harvey, AL, 2208.13804)

Standard cosmology suggests there may have been an early epoch in the universe dominated by potential energy of a scalar field ϕ -> inflation

$$H^2 := \left(\frac{\dot{a}}{a}\right)^2 = \frac{1}{2}\dot{\phi}^2 + V(\phi)$$

If $\dot{\phi}^2/2 \ll V$ (``slow-roll") then $a \simeq a_0 e^{\sqrt{V}t}$ -> exponential expansion

A viable inflationary potential V needs to satisfy a number of requirements:



Determine value/fitness of a model by how well it satisfies these.

An example environment: polynomial potentials

$$V(\phi) \;=\; c_0+c_1\phi+c_2\phi^2+c_3\phi^3+c_4\phi^4+c_5\phi^5+c_6\phi^6$$

coefficient	range	no. of partitions
c_1	$(-10^{-8}, 10^{-8})$	64
c_2	$(-10^{-9}, 10^{-9})$	64
c_3	$(-10^{-10}, 10^{-10})$	64
c_4, c_5, c_6	$(-10^{-11}, 10^{-11})$	64

(III.2)

$$\#\text{states} = 2^{36} \simeq 10^{11}$$

RL does not work well on this environment but GAs do!

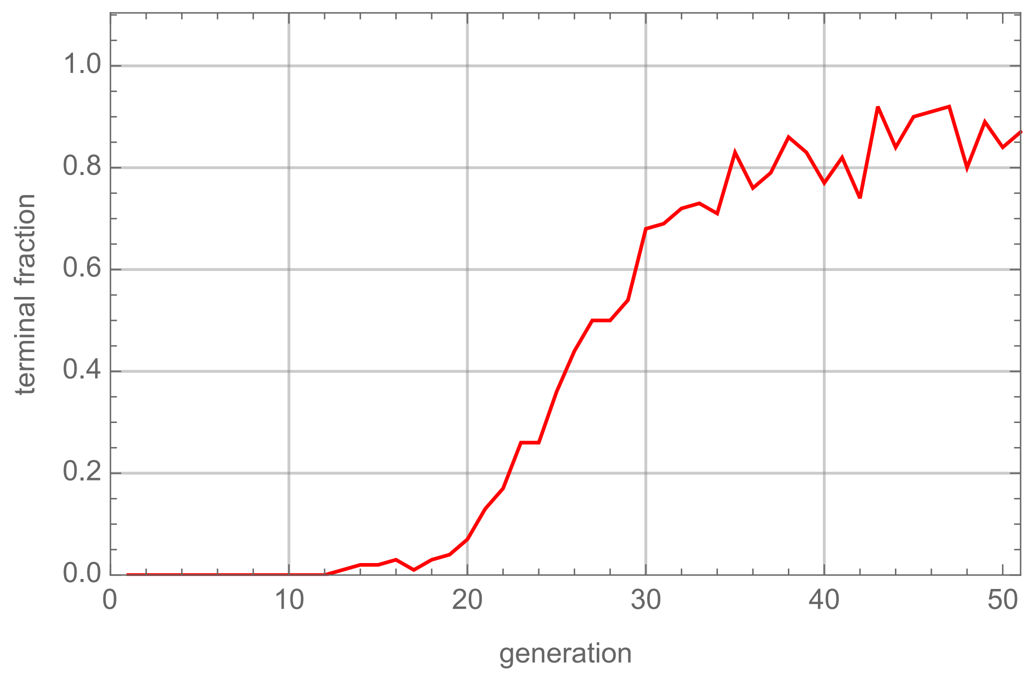


Figure 1. Evolution of the fraction of viable models.

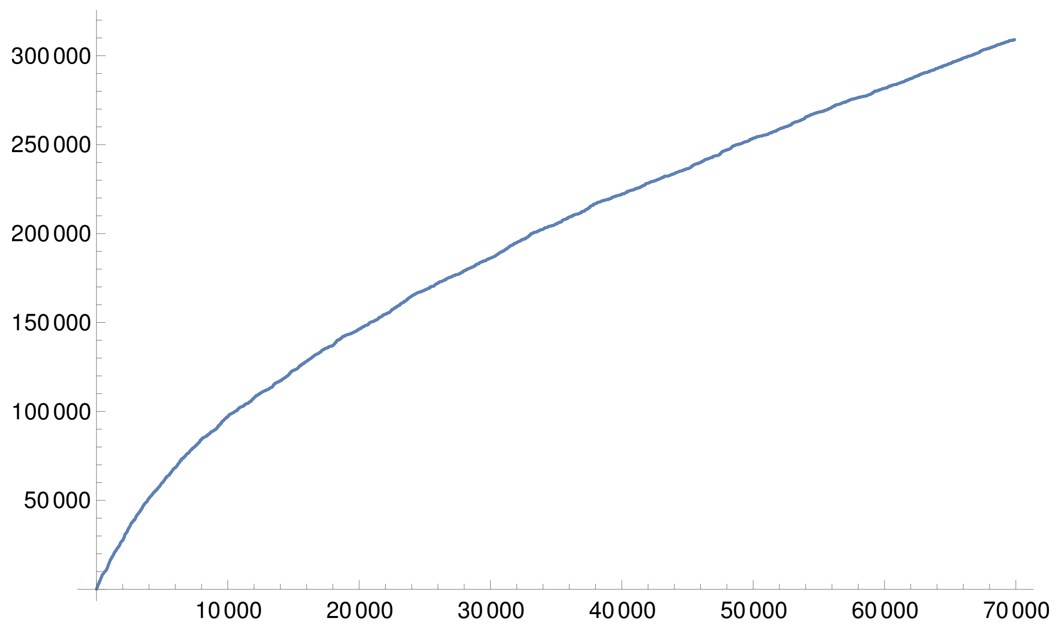


Figure 2. Number of distinct viable models found vs. the number of iterations of GA. Each iteration consisting of 50 generations with a population of 100.

Example models:

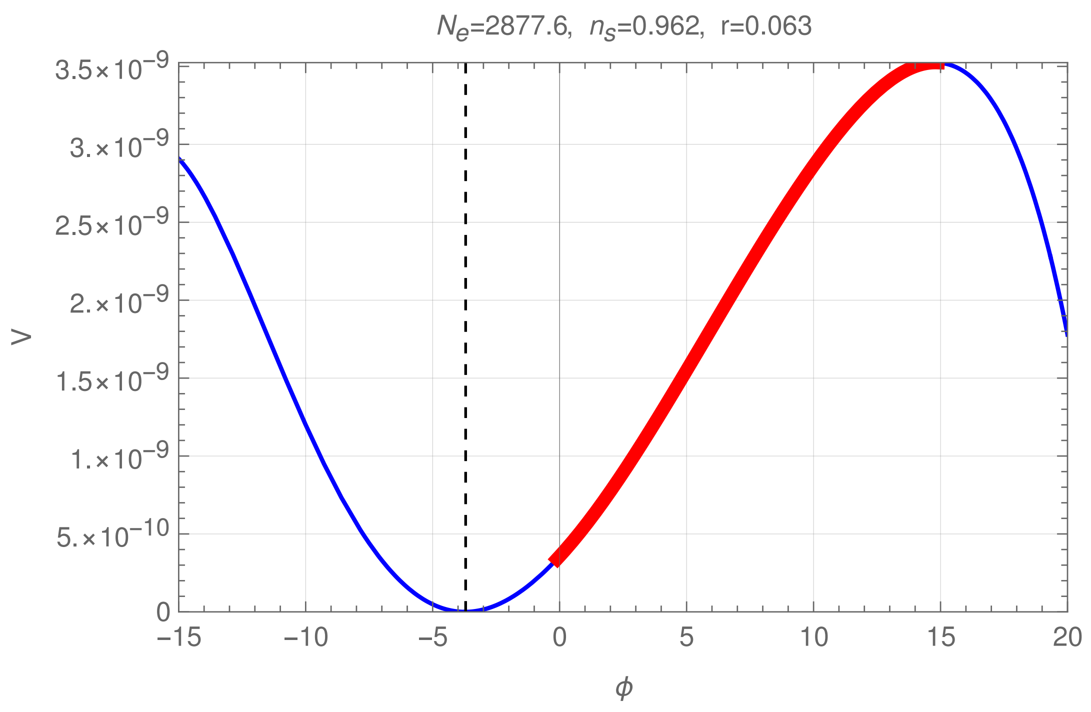


Figure 3. The polynomial potential of degree 6 from Eq. (III.3): large field inflation.

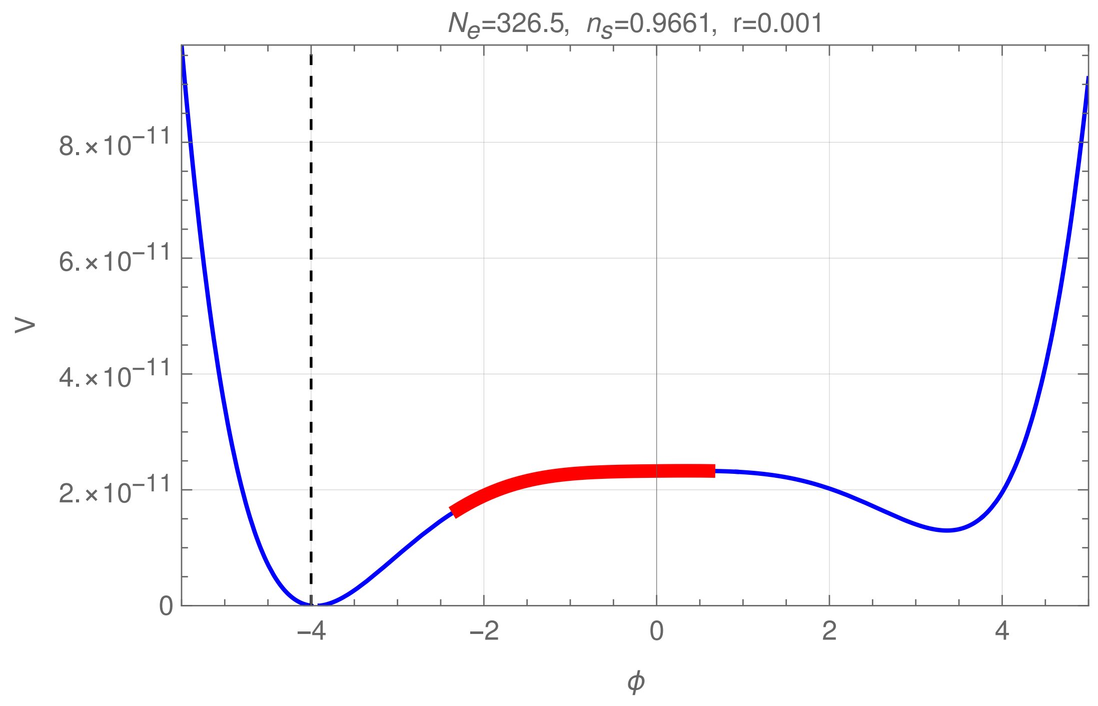


Figure 4. The polynomial potential of degree 6 from Eq. (III.4): small field inflation.

Statistical results:

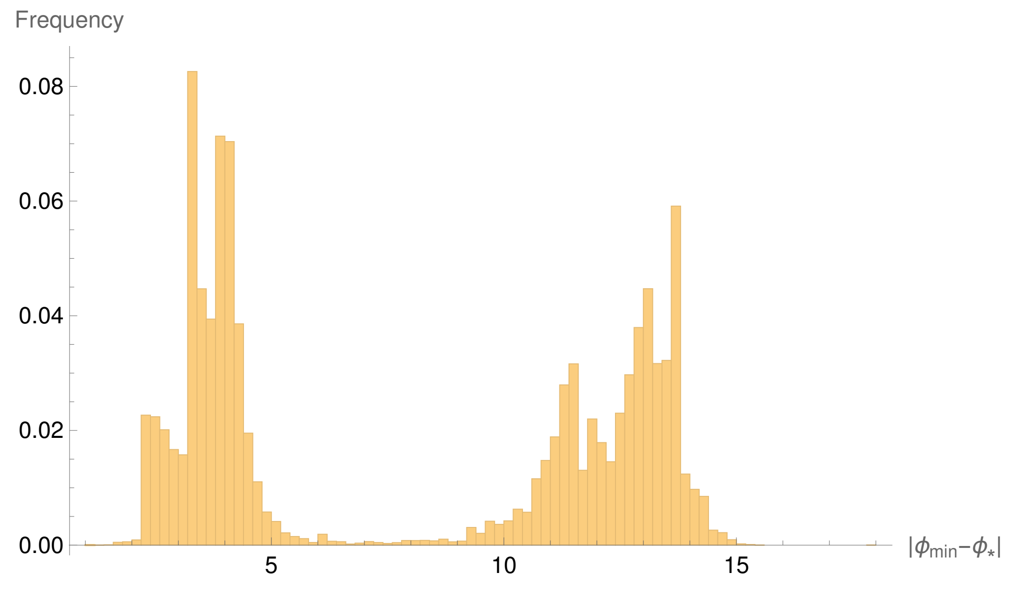


Figure 5. Minimum field excursions during inflation, normalised as a probability distribution (i.e. the sum of all the bars in the histogram is unity).

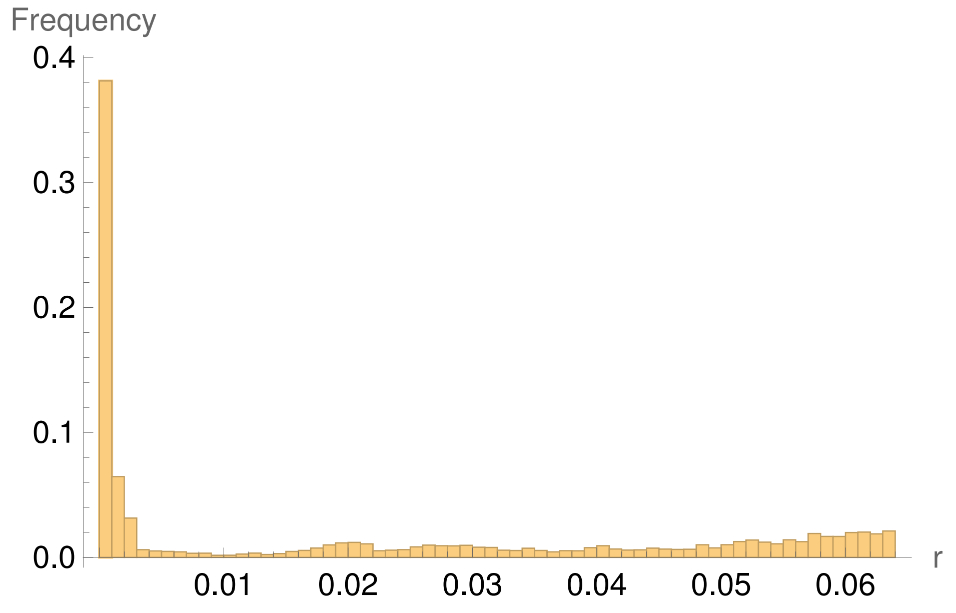


Figure 7. Histogram for the tensor-to-scalar ratio r_*.

Conclusions

- Artificial intelligence/machine learning will dramatically change the way we pursue science.

- There are already many areas of physics where ML has a profound impact.

- I have talked about two: (i) solving differential equations
  (ii) building physical models

- As scientists - working for the common good - we have to forge the way in which these new tools are used, at least in our own field.

감사합니다