MATHEMATICAL DISCOVERY WATCHING A MODEL LEARN

FRANÇOIS CHARTON, AXIOM, ENPC

USAGES OF AI IN MATHEMATICS

- Formalizing and proving
- Coming up with conjectures
- Generating examples

- Building new intuitions about a problem

THE COLLATZ SEQUENCE

Start with a positive integer n

- If it is odd, do $n \to \frac{3n+1}{2}$
- If it is even, do $n \to \frac{n}{2}$

Iterating, you get a sequence, and a famous conjecture (which is not the subject of this talk…)

LONG COLLATZ STEPS

- Start with n odd
- Repeat $n \to \frac{3n+1}{2}$ until you reach an even number (k(n) times)
- Repeat $n \to \frac{n}{2}$ until you reach an odd number (k'(n) times)

- For n=11, we have
  - $11 \to 17 \to 26 \ \ (k(11) = 2)$
  - $26 \to 13 \ \ (k'(11) = 1)$

- Can a transformer learn this function ?

WHY COLLATZ?

AI / CS reasons

- Transformers struggle on arithmetic with large integers
  - Can they learn such a complex arithmetic function?
- Previous results (GCD, modular addition) feature functions with a few outcomes: pattern recognition
  - A harder setting: regression-like, where almost all input values correspond to different outputs
- Deep learning models struggle with compositionality, and complex algorithms
  - A 2-loop algorithm

    $n \to \frac{3n+1}{2}$ repeated k times, k a function of n

    $n \to \frac{n}{2}$ repeated k' times, k' a function of n

WHY COLLATZ?

Mathematical reasons

- A new problem for AI (a welcome change from the "benchmark mentality")
- A provably hard problem: Collatz sequences are known to give rise to complex behavior
- Many theoretical results that we can try to rediscover

Can we learn / rediscover mathematical results just by watching the models learn?

TRANSFORMERS KNOW MORE THAN THEY CAN TELL LEARNING THE COLLATZ SEQUENCE (FC, A. Narayanan, 2025, 2511.10811)

- Generate pairs $(n, \kappa(n))$ for n odd, uniformly distributed from 1 to 10^{12}
  - 5.10^{11} possible pairs, no way the model can memorize them
- Encode them as sequences of digits in base B (all B from 2 to 57)
- Train 56 encoder-decoder transformers (one per base), to predict $\kappa(n)$ from n
  - Dimension 512, 8 attention heads, 4 layers in the encoder, 1 in the decoder
  - Using Int2Int: https://github.com/f-charton/Int2Int
  - Minimizing cross-entropy: no maths involved
- Test the result on 100,000 random examples
  - Success = exact prediction
  - Large problem space, no contamination
  - A new test set at every epoch for further security (ceinture et bretelles)

A WALK IN THE PARK

- Our best models (bases 24 and 32) achieve 99.8% accuracy
  - 250 errors on 100 000 test examples
- Even bases achieve 90+% accuracy
- Odd bases accuracies are lower, and seem to cluster around a few values: 55, 70, 81

Accuracy	Bases
99.5+%	24, 16, 32, 36, 12, 48, 8
99 – 99.5%	4, 18, 56, 40
95 – 99%	6, 20, 28, 2, 52, 10, 44, 54
91 – 95%	42, 22, 30, 14, 50, 34
88 – 89%	26, 46, 33
81 – 82%	9, 45, 38, 15, 17
70 – 71%	49, 47, 21, 27, 51, 39, 31, 41, 23, 7, 25
59 – 66%	57 (65%), 43 (59%)
55 – 56%	13, 37, 19, 55, 29, 53, 35, 5
25 – 37%	11 (37%), 3 (25%)

Table 1: Model accuracy for different bases. Bases listed by decreasing accuracy.

QUANTIZED ACCURACIES

- Accuracy is learned in discrete steps
  - A common pattern in math transformers
- Accuracy levels are quantized, and independent of the base
  - Some bases learn faster, but all go through the same step
  - A pattern not observed in previous experiments (GCD)

- A property of the problem, not of the representation
- Are we learning something about Collatz sequences?

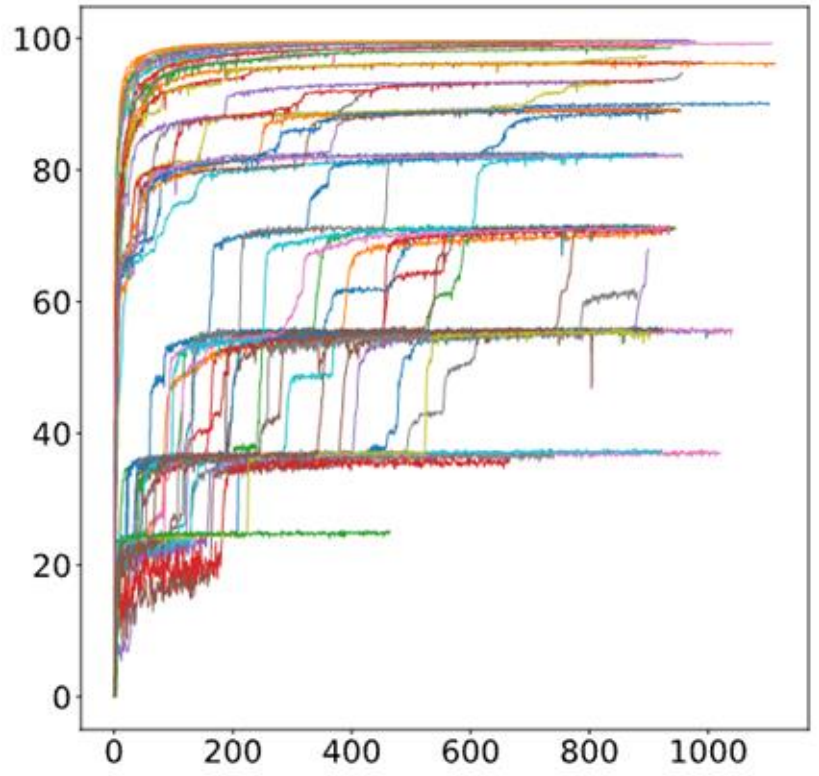


Figure 1: Learning curves for bases 2 to 57 (accuracy vs epochs of 300,000 examples).

WHY THE QUANTIFICATION?

- Quantized steps correspond to special classes of inputs
- The model predict with over 99% accuracy inputs with particular residuals modulo 2^p
  - Inputs with the same binary ending
- And all others with less than 1% accuracy

- These classes form a learning pattern, independent of the base
  - inputs ending in 001 are learned first (for 25% accuracy)
  - then inputs ending in 1011 (for 37.5%)
  - then 1101 (for 50%)
  - …

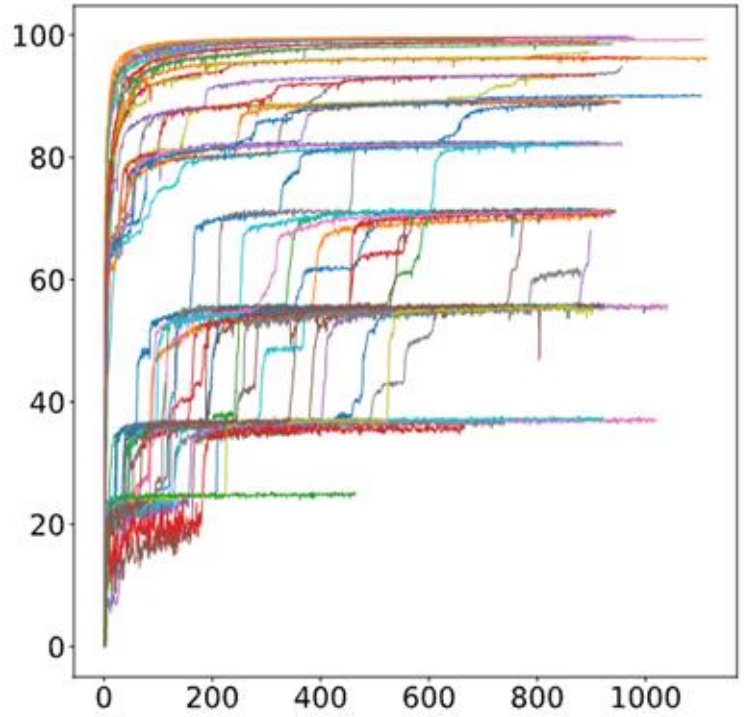


Figure 1: Learning curves for bases 2 to 57 (accuracy vs epochs of 300,000 examples).

WHY THESE RESIDUAL CLASSES?

- All models learn long Collatz step one binary class at a time, in a particular order
  - 001, 1011, 1101, …
  - Once a class is learned, the Collatz mapping is predicted almost perfectly
  - Performing the arithmetic calculations is not a problem

- These classes correspond to special values of k and k', the length of the two loops
  - 001: k=k'=1
  - 1011: k=2, k'=1
  - 1101: k=1, k'=2

A THEOREM ON THE LEARNING PATTERN

- For any odd n, the values of k(n) and k'(n) can be read from the binary representation B_n of n
  - An obvious observation: k is the number of 1s at the right of B_n
  - A much less obvious one: k' is the longest match between B_n and the sequence of parities of $(2^{-p}.(3^k-1) \bmod 3^k)$
  - Related to a theorem of 2006 by Kontorovich and Sinai (Structure theorem for (d,g,h)-maps, ArXiv math/0601622)

Theorem 2 *Let H_l be the sequence of parities of $(a_{l,p})_{0\le p\le \Phi(3^l)}$, written in reverse order:*

$$H_{l,p} = a_{l,\Phi(3^l)-p-1} \mod 2,$$

Let n be an odd positive integer larger than 1, with binary representation B_n. Let 2^l be the largest power of two dividing $n+1$, and S_l the binary sequence $S_l = (H_l)^m 1^l$, with m chosen so that $|S_l| > |B_n|$, then

i. B_n shares with S_l a common suffix of length $s \ge l+1$,
ii. the long Collatz step of n verifies $k = l$ and $k' = s - l$.

THE LEARNING PATTERN

- The model learns the long Collatz step one pair (k,k') at a time
  - In increasing order of k+k'
- Once a pair is learned, all inputs associated with k and k' (with a certain binary ending) are correctly predicted
- Other inputs are predicted wrong

- All models are learning the mapping $n \rightarrow (3^k (n+1) / 2^k - 1) / 2^{k'}$ for particular values of k and k'
- Not learning the function $\kappa(n)$, but its restriction for special classes of n
  - A special kind of "universal approximators"

UNDERSTANDING ERRORS

DOES THE MODEL HALLUCINATE?

HALLUCINATION IN LANGUAGE MODELS

- A documented phenomenon
  - Language models fail at random
  - Make irrelevant prediction

- Damning for math / science transformers
  - We want principled errors, that happen for particular input classes
  - And wrong predictions that are close to correct values

INVESTIGATING ERRORS

- Select a trained model
- Run it on 100,000 random test cases (not seen during training, but in-domain)
- Investigate errors, by comparing wrong predictions to correct values
- Study the ratio r = p/t : p predicted value, t correct value of the long Collatz successor

ERRORS ARE SHOCKINGLY NONRANDOM
MORE QUANTIZATION

- The distribution of the ratios p/t is discrete, and takes values of the form: $a.2^n$ with a = 2, 2/3, 4/9, 2/B…
- A sum of power laws

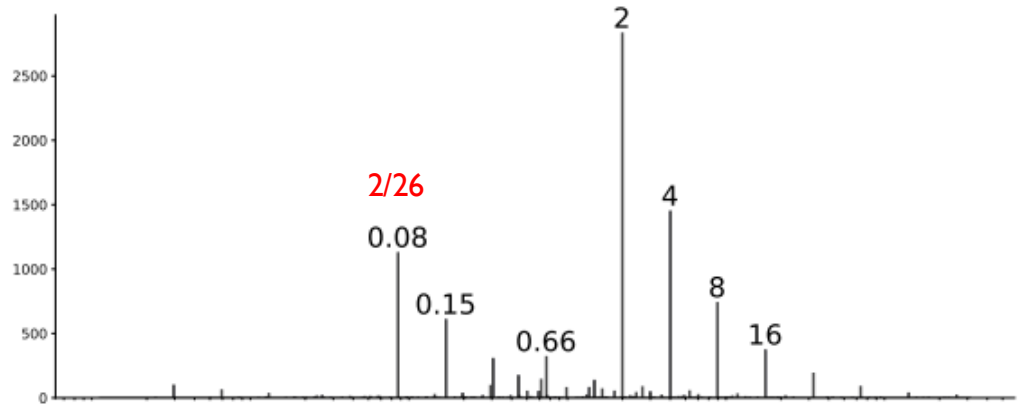


Figure 2: Distribution of p/t: base 26 (all model errors)

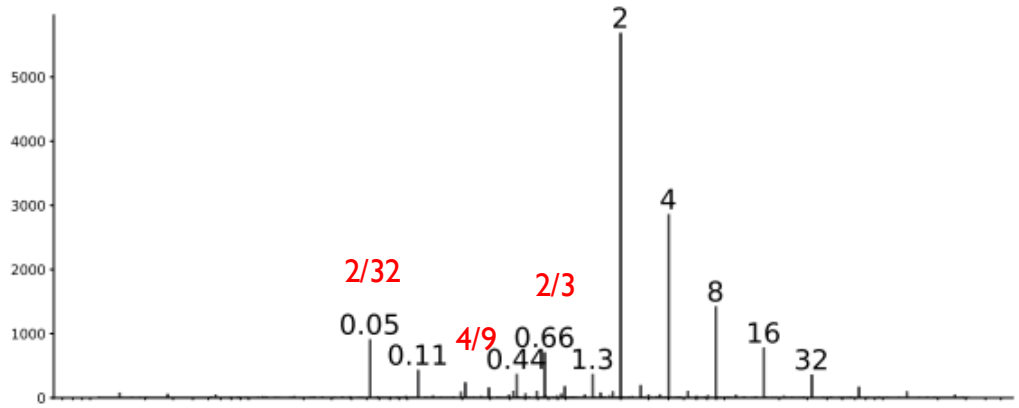


Figure 3: Distribution of p/t: base 38 (all model errors)

ERRORS IN OUR BEST MODELS

- Base 24: 243 errors out of 100,000 test examples,

- Errors happen for specific input classes: large values of k and k'
  - 210 errors (out of 243) have k≥8,
  - the model always fails when k≥11
  - k=9 and k'=1: 41 correct / 14 wrong
  - k=9 and k'=2: 5 correct / 26 wrong
  - k=9 and k'>2; all model predictions are wrong
- A consequence of the learning pattern

ERRORS IN OUR BEST MODELS

- Incorrect predictions are close to the correct values : the ratio r = p/t is close to 1
- In base 24
  - 111 errors (46%) are within 0.1% of the correct value,
  - 204 (84%) within 1%,
  - 225 (93%) within 3%.
- In base 32
  - 47% within 0.1%
  - 93% within 1%
  - 98% within 3%

ERROR IN OUR BEST MODELS

- In fact, model predictions and correct solutions, encoded in the base, share more than half their tokens
  - Correct solution (base 24) [4, 21, 20, 6, 7, 3, 20, 2, 20, 1],
  - Model prediction [4, 21, 20, 6, 7, 8, 8, 2, 20, 1]
  - The prediction is correct modulo 24^3,

	Errors	Prediction length	Correct tokens	Correct prefix	Correct suffix
Base 24	243	9.4	4.5	2.1	2.3
Base 32	265	8.9	4.2	1.9	2.1
Base 16	259	11.1	5.8	2.7	2.7
Base 8	479	14.5	8.0	3.5	3.6
Base 48	399	7.7	4.0	1.9	2.0
Base 36	287	7.7	3.3	1.5	1.6
Base 12	315	11.7	6.1	2.6	2.9

Table 2: Prediction errors for different good bases. Average length of predictions, number of tokens in agreement between prediction and target.

ERRORS IN OUR BEST MODELS

- The models do not confabulate
- Incorrect predictions are close to the correct value: in absolute value, and modulo B^p

- The models do better than their accuracy suggests

EVEN PREDICTIONS

- All inputs are odd, and outputs should be odd
  - This is observed throughout the training set
- In even bases, less than 0.8% of model predictions are even (almost zero for the best bases)
- Less than 5% of model errors are even: errors have the correct parity

- In odd bases, in 85 to 90% of model errors, an even number is predicted, way more than chance!

POWER OF 2 ERRORS

- In odd bases, in 70 to 80% of errors, the model is **exactly wrong** by a small power of two: it predicts p=2t, p=4t or p=8t
- The model learns the correct k, but underestimates k'
- This happens for specific values of k and k'
  - Only for values of k where the model can predict k'=1,
  - k' is then predicted as the largest correct value

AN EXAMPLE IN BASE 27

- 71% accuracy, power of two errors account of 75.5% of model errors
- The model correctly predicts inputs with
  - k=1 and k'=1,2,3,
  - k=2 and k'=1, 2,
  - k=3 and 4, for k'=1
- Power of two errors happen for
  - For k=1, k'>3. The model uses k'=3 (largest correct value)
    - For k=1, k'=6, the model will predict 8 $\kappa(n)$, for k'=4 the model predicts 2 $\kappa(n)$
  - For k=2, k'>2. The model uses k'=2
  - For k=3 and 4, k'>1. The model uses k'=1

POWER OF TWO ERRORS IN EVEN BASES

- A similar situation occurs in even bases
- But the model insists on predicting an odd number
  - A bias learned during training
- p = 2t + e, with |e| = 1, or $B/2^p$
- In some cases, these errors are "one token short", the model predicts $r \approx \frac{2^p}{B}$

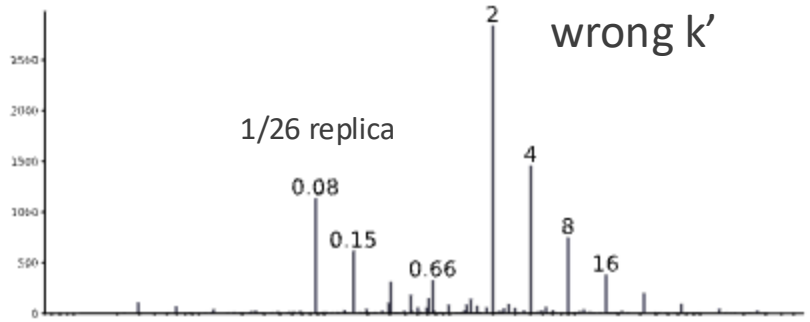


Figure 2: Distribution of p/t: base 26 (all model errors)

OTHER ERRORS

- In most other cases, the error is still quantized
- The model predicts a smaller value of k, and k'=1

- In fact, the model predict the largest value of k it can correctly predict, and k'=1
- These errors, together with power of two errors, account for more than 90% of model errors

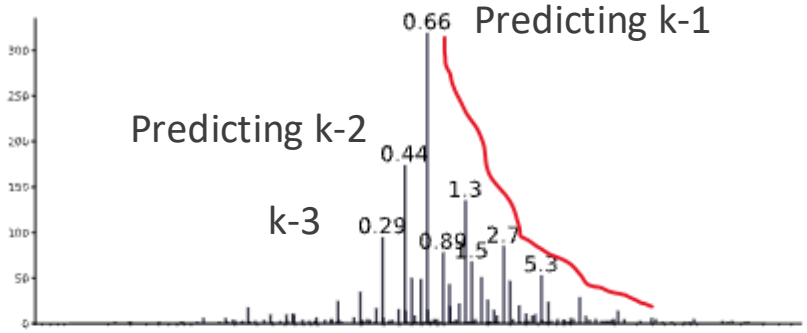


Figure 4: Distribution of p/t: base 26 (all model errors, excluding near power of two)

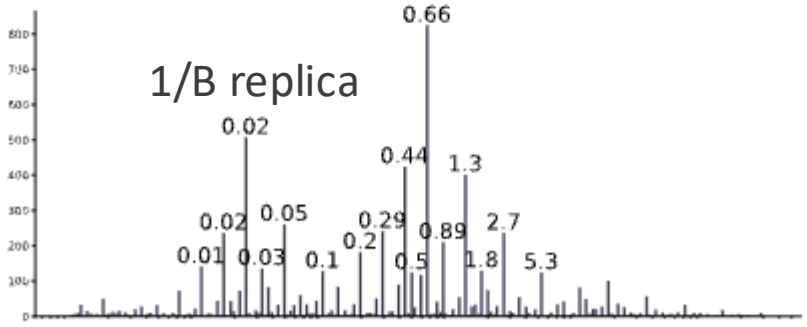


Figure 6: Distribution of p/t: base 27 (all model errors, excluding power of two)

THE ERROR PATTERN

- For over 90% of errors, we have two main cases, depending on the value of k(n)
- Let k_{max} be the largest value that the model can predict
- If $k \leq k_{max}$, the model will predict as if k'=l', the largest value of k' that the model can predict correctly for k
- Else, it will predict (k_{max}, l)

A GENERALIZED LEARNING PATTERN

- Models learn to classify inputs with particular values of k and k'
  - from the binary representation of n
  - this is why even bases perform better
- At any point in training, k is learned up to a value k_{max} and for each k, k' up to a value $k'_{k,max}$
- All input with $k \leq k_{max}$ and $k' \geq k'_{k,max}$ are predicted as $f(k,k'_{k,max})$,
- All inputs with $k > k_{max}$ are predicted as $f(k_{max}, 1)$

- This accounts for more than 90% or errors, in all bases, throughout training

LESSON LEARNED – MATH MINING

- Complex arithmetic functions can be learned by supervised transformers
- Watching models learn can tell us a lot about the underlying mathematics
- Model errors may even prove insightful

- Can this be a tool for mathematical insight?
- Can we make this more systematic?
- How should we design such experiments?

LESSON LEARNED – THE TACIT DIMENSION

- Close and principled model failures indicate that the transformer knows more about the problem than its decoded output tell us
  - Tacit knowledge (Polanyi)
- The $d.\log_B(n)$-dimensional representation space used to encode inputs holds rich information,
  - Can we probe it?
  - Can we design geometric methods to recover such information?
- Should we design architecture / experiments that foster the learning of such rich representations
  - Math world models
  - Better decoders (as Yann LeCun has been advocating)
  - Representation learning

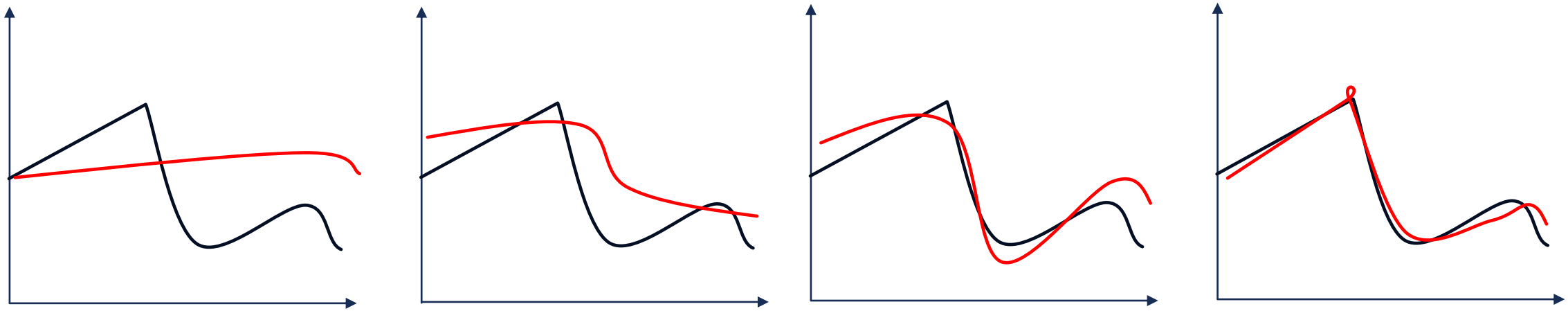
LESSONS LEARNED – MATHS FOR AI

- Math problems can be used to study how models learn
  - Leveraging our knowledge of the underlying problem, and oour understanding of the distributions of mathematical objects
- Here, we could elucidate the algorithm learned by the transformer
- And throw hallucination out of the picture
  - Because of supervision?

- Can well-understood math problems be the new toolbox for AI interpretability research?

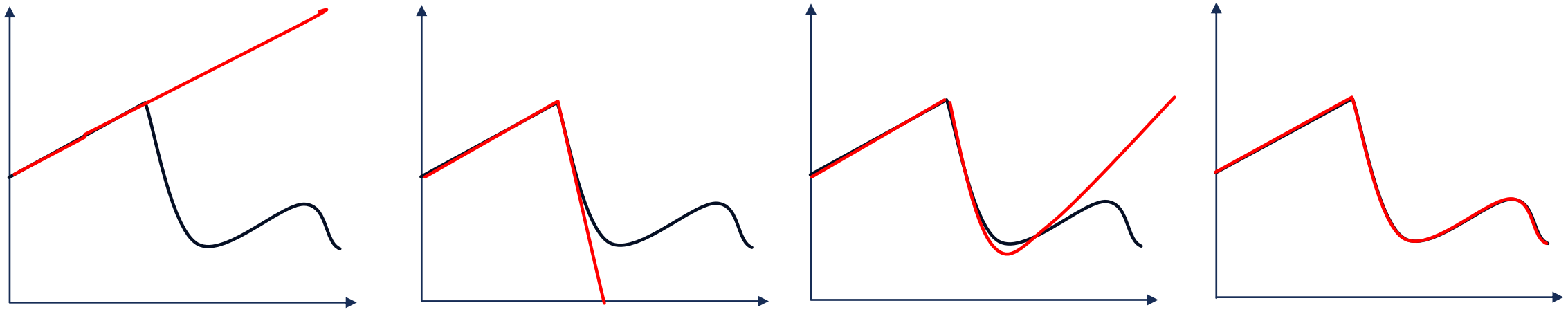
LESSON LEARNED - UNIVERSAL APPROXIMATORS, REALLY?

- It is known (Cybenko, Hornik, 1989) that neural network are universal approximators
- Given enough examples, they can learn to minimize any loss, to approximate any function
- Intuitively, we imagine successive approximations to proceed like so



LESSON LEARNED - UNIVERSAL APPROXIMATORS, REALLY?

- For the Collatz long step, successive approximations proceed in a different way



LESSON LEARNED - UNIVERSAL APPROXIMATORS, REALLY?

- The model is not learning the full algorithm, or the complete function, but a series of special cases
- For each case, approximation is almost perfect
- This points to a two-step process, observed (retrospectively) in previous works:
  - Input space is clustered into a small number of classes (according to their binary endings, here)
  - A specific function is learned on each class (this is the easy part, even when the function is a complex arithmetic function)
  - Pattern matching over inputs, then regression to outputs
- Future challenges:
  - Understanding how this pattern emerges
  - How it relates to data distribution
  - Whether it can be controlled

SUGGESTED READINGS

- Learning the Euler factors of elliptic curves, Babei et al 2025 (arXiv:2502.10357).
- Interpreting Machine Learning to Build a Zeta Map Algorithm, Huang, et al. 2025, (arXiv:2511.124210).