Kolmogorov Complexity and How it Illuminates our Limitations to Let Machines Learn Recursion

by

Lukas J. Rüttgers

Submitted to the Department of Computer Science
in partial fulfillment of the requirements for the degree of

BACHELOR OF SCIENCE IN COMPUTER SCIENCE

at the

RWTH AACHEN UNIVERSITY

September 2024

Authored by: Lukas J. Rüttgers
Department of Computer Science
June 30, 2024

Certified by: Jingzhao Zhang
Assistant Professor of Computer Science, Tsinghua University, Thesis Supervisor

Certified by: Hector Geffner
Professor of Computer Science, RWTH Aachen University, Thesis Supervisor

Kolmogorov Complexity and How it Illuminates our Limitations to Let Machines Learn Recursion

by

Lukas J. Rüttgers

Submitted to the Department of Computer Science
on June 30, 2024 in partial fulfillment of the requirements for the degree of

BACHELOR OF SCIENCE IN COMPUTER SCIENCE

ABSTRACT

Out-of-distribution generalization requires agents to induce from their experience to unseen environments. While recursion is a powerful tool for compressing algorithmic behaviour, the hardness of learning recursive descriptions has impeded its integration into modern machine learning pipelines.

This work puts forward a new complexity measure for algorithmic descriptions that captures the informational "simplicity" of a program and is based on Kolmogorov complexity. Intuitively, this measure characterizes the complexity of boolean functions by the minimum functional information required to ensure that the simplest algorithm that is consistent with the given information indeed computes the desired function.

With this measure at hand, it emphasizes the efficacy of learning recursive algorithmic descriptions for reasonable behaviour outside of the training distribution, and showcases drawbacks in existing frequently employed models and optimization objectives in capturing elementary recursive patterns.

Thesis supervisor: Jingzhao Zhang
Title: Assistant Professor of Computer Science, Tsinghua University

Thesis supervisor: Hector Geffner
Title: Professor of Computer Science, RWTH Aachen University

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 The i.i.d. Assumption

1.2 Domain Generalization

1.3 Outline of This Work

1.3.1 Purpose and Contributions

1.3.2 Content Organization

Chapter 2

Domain Generalization of Neural Networks

2.1 Extrapolation Drawbacks of Models

2.1.1 ReLU MLPs Extrapolate Linearly

Within the support of the training distribution, MLPs are universal function approximators. In the NTK regime, ReLU MLPs converge to linear functions outside the training distribution with a linear convergence rate. Instead, the non-linearities in the architecture are the crucial foundation for encoding task-specific non-linearities. Compare Graph Neural Networks and Dynamic Programming Problems.

2.2 Invariant Causal Mechanisms

2.3 Invariant Risk Minimization

2.3.1 Fully Informative Invariant Features

But I argue that there is a far larger drawback.

2.3.2 Learning Prime Numbers

Consider the decision problem PrimeNumbers. It is widely believed that there is no efficient algorithm that decides prime numbers.

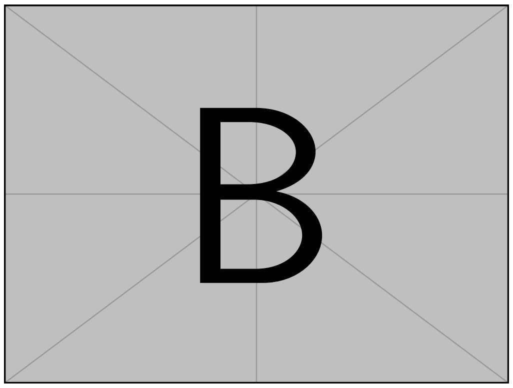

Figure 2.1: This figure is yet a placeholder and is supposed to showcase experimental visualizations of the immediate extrapolation behaviour of ReLU MLPs trained with L2 Regularization in experiments.

2.4 Other Optimization Objective Reformulations

2.4.1 Risk Extrapolation

2.4.2 Distributionally Robust Optimization

Empirically, none of the above algorithms perform better than ERM.

Chapter 3

Recursion and Simplicity

3.1 Kolmogorov Complexity

Denote by $\mathrm{enc}(\mathcal{T})$ the Gödel number of a Turing Machine.

Definition 3.1 (Kolmogorov Complexity)

3.1.1 The Implicit Bias within Kolmogorov Complexity

Theorem 3.2 (Invariance Theorem)
There is only an additive constant between Kolmogorov Complexities of any pair of universal Turing Machines.

In the following, we assume an arbitrary, fixed, finite alphabet Σ of cardinality $|\Sigma| =: r$. Without loss of generality, we identify Σ with the field $\mathbb{Z}_r$ to which it possesses a natural isomorphism π. When we speak of $0, 1 \in Sigma$, we refer to the elements $a, b \in \Sigma$ with $\pi(a) = 0, \pi(b) = 1$. Since $r \in \mathbb{N}$ is finite, $\mathfrak{C} \cong \mathbb{Z}_r$ is equivalent to $\mathfrak{C} \equiv \mathbb{Z}_r$ for any mathematical structure $\mathfrak{C}$. That is, $\mathfrak{C}$ will satisfy exactly those first-order logic formulas that $\mathbb{Z}_r$ satisfies. In our case, this also holds for second-order logic formulas formulated over $\mathbb{Z}_r$, as quantification over predicates and relations over finite objects is expressible in first-order logic, too.

Definition 3.3 (Universal Turing Machine)
We define a universal Turing Machine U that receives both program index and input in a self-delimiting encoding. The construction is in the spirit of [3, Section 2.1] as follows. U expects input in the form 1^n0xy, where $x, y \in \{0, 1\}^*$ and $l(x) = n$. U hence interprets the number of leading 1s as the length of the first input. It can uniquely identify the number of leading 1s by the first appearance of 0 on the input tape. U interprets x as the encoding of the index of the Turing Machine it shall execute and retrieves the Gödel number of this Turing Machine in the natural way. That is, U incrementally iterates through all binary strings, checks whether they encode a Gödel number and subtracts x by one each time it encounters a valid Gödel number. When x has been reduced to the empty string, the current binary string will encode the n_xth valid Gödel number of some Turing Machine $\mathcal{T}_x$.

Then, U simulates $\mathcal{T}_x$ on input y.

Definition 3.4 (Strings with Infinite Zero Padding)
For any $x \in \Sigma^*$, we define $x\overline{0} \in \Sigma^\infty$ as the infinite string with prefix x and infinitely many subsequent 0*s.*

Lemma 3.5 (Additional Complexity of Infinite Zero Padding)
There is a constant c_{pad} such that for every string $x \in \Sigma^*$, $K(x\overline{0}) \leq K(x) + c_{pad}$.

Proof. We construct a Turing Machine $\mathcal{T}_{pad}$ in the following way. $\mathcal{T}_{pad}$ always moves its tape head to the right. It leaves symbols $a \in \Sigma$ unchanged and replaces every blank symbol B by 0. For any input x, $\mathcal{T}_{pad}(x) = x\overline{0}$. This Turing Machine is well-defined. In particular, it has a finite Gödel number $\text{enc}(\mathcal{T}_{pad})$ of length $c_{enc} \in \mathbb{N}$. There is an $p \in \mathbb{N}$ such that $\text{enc}(\mathcal{T}_{pad})$ is the pth Gödel number in the natural enumeration of Gödel numbers that is also computed by our universal Turing Machine U from Definition 3.3. Therefore, there exists a constant $c_0 \leq \log_2(p)$ such that $K(\mathcal{T}) = c_0$.

As providing x_p and x to the universal Turing Machine U in the self-delimiting format $1^{c_0}0x_px$ yields $x\overline{0}$, we finally obtain $K(x\overline{0}) \leq \underbrace{2c_0 + 1}{} =: c_{pad} + K(x)$. □

3.1.2 Compressibility

Lemma 3.6 (Incompressible Strings)
For any $n \in \mathbb{N}$, there exists a string $v \in \Sigma^n$ with $K(v) \geq l(v) = n$.

Proof. Let $r := |\Sigma|$, thus $|\Sigma^n| = |\Sigma|^n = r^n$. All these strings are different objects and consequently must have different encodings. For any $v, w \in \Sigma^n$, if we have $v \neq w$ and $U(p) = v$ and $U(q) = w$ for some universal reference machine U and programs $p, q \in \Sigma^*$, then necessarily $p \neq q$. But the geometric sum $\sum_{i=0}^{n-1} r^i = r^n - 1$, hence there are only $r^n - 1$ strings in Σ^* with length at most $n - 1$. By the pigeonhole principle, there must be at least one $v \in \Sigma^n$ such that there exists no program p of length $l(p) < n$ with $U(p) = v$. For this v, we have $K(v) \geq n = l(v)$. □

We extend the above lemma to infinite strings with finite hamming weight.

Corollary 3.6.1 (Incompressible Strings with Infinite Paddings)
For any $n \in \mathbb{N}$, there exists a string $v_n = x_n\overline{0} \in \Sigma^\infty$ with $K(v_n) \geq l(x_n) = n$.

Proof. The argument is analogous to Lemma 3.6. For any $n \in \mathbb{N}$, there are exactly 2^n strings $v \in \Sigma^\infty$ that satisfy $v = x\overline{0}$ for some $x \in \Sigma^*$. However, there are only $2^n - 1$ strings of length shorter than n that could serve as encodings for strings like v. Therefore one of the 2^n strings is not compressible beyond length n. □

Lemma 3.7 (Logarithmically Compressible Strings)
There exists a constant $c \in \mathbb{N}$ such that: For every $n \in \mathbb{N}$, there exists a string $z_n \in \Sigma^*$ of length $l(z_n) = n$ with Kolmogorov complexity $K(z_n) \leq \log(n) + c$.

Proof. Without loss of generality, $0, 1 \in \Sigma$. Since Σ is finite, it is isomorphic $\{0, 1, \ldots, r-1\}$, where $r = |\Sigma|$. We construct a Turing Machine $\mathcal{T}$ as follows: If the string x on the input tape does not only consist of 0s and 1s, $\mathcal{T}$ immediately terminates and hence outputs ε. In the following, we assume $x \in \{0,1\}^*$. $\mathcal{T}$ interprets x as the encoding of a natural number in the usual way. Leading zeros are ignored and immediately replaced by the blank symbol B. We denote $n_x \in \mathbb{N}$ as the natural number encoded by x. Conversely, for any $n \in \mathbb{N}$, x_n denotes the unique binary string encoding it without leading zeros. Accordingly, 0 is encoded by the empty string ε. $\mathcal{T}$ now repeats the following procedure until the input tape only holds the empty string ε.

- Write a 1 on the next free box.
- Subtract the input by 1.

Finally, $\mathcal{T}$ returns a string of n_x subsequent 1s. In particular, the partial computable function f computed by $\mathcal{T}$ satisfies $f(n_x) = 1^n$ for each $n \in \mathbb{N}$. Analogously as in the proof of Lemma 3.5, there exists a string x_p and a constant $c_\mathcal{T} \leq \log_2(p), p \in \mathbb{N}$, such that $K(\mathcal{T}) = c_\mathcal{T}$.

For any $n \in \mathbb{N}$, providing $1^{c_\mathcal{T}}0x_px_n$ to the universal Turing Machine U from Definition 3.3 as input yields $z_n := 1^n$ as output. For that reason, $K(z_n) \leq 2c_\mathcal{T} + 1 + l(x_n) \leq \underbrace{2c_\mathcal{T} + 2}_{=:c} + \log(n)$.

□

Corollary 3.7.1 (Log-Compressible Strings with Infinite Zero Padding)
There exists a constant $c \in \mathbb{N}$ such that: For every $n \in \mathbb{N}$, there exists a string $p_n \in \Sigma^\infty$ such that $p_n = z_n\overline{0}$ for a string z_n of length $l(z_n) = n$ and that has Kolmogorov complexity $K(p_n) \leq \log(n) + c$.

Proof. By virtue of Lemma 3.5, padding the strings of the form 1^n with infinite 0s required only a constant additional amount c_{pad} of descriptive information. Combined with the constant c from the proof of Lemma 3.7, we obtain $K(p_n) \leq \log(n) + c + c_{pad}$ for $p_n = 1^n\overline{0}$. □

Lemma 3.8 (Arbitrarily Compressible Strings)
There exists a constant $c \in \mathbb{N}$ such that: For every $n \in \mathbb{N}$, there exists a string $z_n \in \Sigma$ of length $l(z_n) = \left(\bigcirc_{i=1}^n \exp_2\right)(1)$ with Kolmogorov complexity $K(z_n) \leq \log(n) + c$.

Proof. This time, we construct a Turing Machine $\mathcal{T}$ that computes an exponential tower function. The function f computed by $\mathcal{T}$ will satisfy $f(x_n) = \begin{cases} 1^{\left(\bigcirc_{i=1}^n \exp_2\right)(1)}, & n \in \mathbb{N}, n \geq 1 \\ 1, & n = 0 \in \mathbb{N}. \end{cases}$

This is done in the following way: Just as in the proof of Lemma 3.7, $\mathcal{T}$ interprets the input x as the encoding of a natural number n_x. Besides the input tape and the output tape, $\mathcal{T}$ uses one auxiliary tape. If x does not satisfy the expected format, $\mathcal{T}$ immediately terminates. Otherwise, $\mathcal{T}$ first writes a 1 on its output tape. Now, it repeats the following procedure while the string on the input tape is not eradicated blank.

- Remove all non-blank symbols on the auxiliary tape.
- Copy the non-blank symbol sequence x on the output tape to the auxiliary tape.
- Interpret x as the natural number $n_x \in \mathbb{N}$. Compute $x_{2^{n_x}}$ just as in the proof for Lemma 3.7 and write it on the output tape.

- Subtract the number on the input tape by 1.

The string x on the output tape now represents $n_x = \left(\bigcirc_{i=1}^{n} \exp_2\right)(1)$. Finally, copy x once more to the auxiliary tape and clean the output tape. Now, repeat the following procedure as long as the string on the auxiliary tape is not eradicated blank.

- Write a 1 on the output tape.

- Subtract the string on the auxiliary tape by 1.

With $1^{\left(\bigcirc_{i=1}^{n} \exp_2\right)(1)}$ written to the output tape, $\mathcal{T}$ terminates.

By the same argument as in the proof of Lemma 3.7, there is a program string p that identifies $\mathcal{T}$ with a length of $c_{\mathcal{T}} \in \mathbb{N}$. For any n, providing $1^{c_{\mathcal{T}}}0px_n$ to the universal Turing Machine U yields the output $z_n := 1^{\left(\bigcirc_{i=1}^{n} \exp_2\right)(1)}$ of length $\left(\bigcirc_{i=1}^{n} \exp_2\right)(1)$. Henceforth, $K(z_n) \leq \log(n) + c$ for $c := 2c_{\mathcal{T}} + 2$.

$\square$

Corollary 3.8.1 (Arbitrarily Compressible Strings with Infinite Zero Padding)
There exists a constant $c \in \mathbb{N}$ such that: For every $n \in \mathbb{N}$, there exists a string $p_n \in \Sigma^\infty$ such that $p_n = z_n\overline{0}$ for a string z_n of length $l(z_n) = \left(\bigcirc_{i=1}^{n} \exp_2\right)(1)$ and that has Kolmogorov complexity $K(p_n) \leq \log(n) + c$.

Proof. The proof works just as in Corollary 3.7.1 and merely requires replacing c and p_n by the respective values in the proof of Lemma 3.8. $\square$

3.2 Regularization and Kolmogorov Complexity

Definition 3.9 (Turing Machine for Permutation)
Let π be an arbitrary permutation over Σ. We define $\mathcal{T}_\pi$ in the straightforward way. $\mathcal{T}$ goes over the input and replaces every symbol $a \in \Sigma$ by $\pi(a)$. After having arrived on the right end, it moves back to the first symbol and terminates.

δ	0	1	$\dots$	$r-1$	B
q_0	$(q_0, \pi(0), R)$	$(q_0, \pi(1), R)$	$\dots$	$(q_0, \pi(r-1), R)$	(q_2, B, L)
q_2	$(q_2, 0, L)$	$(q_2, 1, L)$	$\dots$	$(q_2, r-1, L)$	(q_1, B, R)

Table 3.1: The schematic transition function δ of $\mathcal{T}_\pi$. By convention, q_1 represents the final state and is hence omitted.

It is trivial to show that the length of the Gödel number of $\mathcal{T}_\pi$ is the same for all permutations π over Σ, namely $c := l(\mathcal{T}_\pi) = a \cdot |\Sigma| + b$ for small constants $a, b \in \mathbb{N}$. The interested reader is referred to Lemma A.3 in the Appendix.

As is rigorously proved thereafter in Lemma A.4, the Turing Machine $\mathcal{T}_\pi$ from Definition 3.9 is optimal in terms of its Gödel number length and therefore also in terms of the

Kolmogorov complexity of π. For any permutation π except the identity function, there is no Turing Machine $\mathcal{T}$ that computes π and achieves Gödel number of shorter length than $\mathcal{T}_\pi$. Moreover, this optimal length is invariant across all permutations over Σ - except the identity function. With Lemma A.3, the interested reader will also find a tight bound for the length of $\mathrm{enc}(\mathcal{T})_\pi$.

Theorem 3.10 (Kolmogorov Complexity Invariance under Permutation)
There exists a small constant c_Σ that scales only quadratically with $|\Sigma|$ such that the following holds: Let π be an arbitrary permutation over Σ that is not equivalent to the identity function. Let $x \in \Sigma^*$ be an arbitrary string over Σ. Then $|K(x) - K(\pi(x))| \leq c_\Sigma$.

Proof. Since π is a permutation, it has an inverse function π^{-1}. Since π is not equivalent to the identity function, π^{-1} is neither. By Lemma A.3, $K(\pi), K(\pi^{-1}) \leq c_0 := 2|\Sigma|^2+31|\Sigma|+34$.

Let $x \in \Sigma^*$ be arbitrary. Denote by $\pi(x) \in \Sigma^*$ the string that is obtained by applying π to every symbol in x.

Then, providing $z = 1^{c_0}0\,\mathrm{enc}(\mathcal{T}_\pi)x$ to the universal Turing Machine U from Definition 3.3 yields $U(z) = \pi(x)$. Consequently, $K(\pi(x)) \leq 2 \cdot c_0 + 1 + K(x)$.

Since the above argument holds for any string $x \in \Sigma^*$, it also holds for $\pi^{-1}(x)$ for any $x \in \Sigma^*$. Therefore, we symmetrically have $K(x) \leq 2 \cdot c_0 + 1K(\pi^{-1}(x))$ and hence $K(x) \leq 2 \cdot c_0 + 1K(\pi(x))$. Stitching these two bounds together and choosing $c_\Sigma := 2c_0 + 1$, we finally obtain the desired result. □

By elementary calculus, the following result is established. The proof is omitted here and laid out in Appendix A.1 instead.

Lemma 3.11 (Log-Linear Inequality with Additive Constant)
Let $a \geq 1, b \geq 0$ be arbitrary real numbers. For any real number $x > 2^{4(a+b)}$, it holds that $a\log_2(x) + b < x$.

Definition 3.12 (Properly additive vector norms)
Let $\|\cdot\| : \Sigma^\infty \to \mathbb{R}_{\geq 0}$ be an arbitrary norm over Σ^∞. Denote by $e_1, e_2, \ldots$ the unit vectors that form an orthogonal basis of Σ^∞. That is, e_i consists only out of 0s except for index i, where it carries a 1.

We say that $\|\cdot\|$ is properly additive if for any proper subset $A \subset \mathbb{N}$ and $v_A := \sum_{i \in A} e_i$, and any $j \notin A$, it holds that

$$\|v_A + e_j\| > \|v_A\|. \tag{3.1}$$

Vector norms that do not satisfy this condition are called improperly additive.

With this lemma an definition at hand, we now show the following.

Theorem 3.13 (Unbounded Order Inconsistencies to any Vector Norm)
Let $\|\cdot\|$ denote an arbitrary properly additive norm over Σ^∞ as by Definition 3.12. For any pair of real numbers $a \geq 1, b \geq 0$, there are $v, w \in \Sigma^\infty$ such that $K(v) \geq a \cdot K(w) + b$, but $\|v\| < \|w\|$.

Proof. Let $a \geq 1, b \geq 0$ be arbitrary real numbers. Let $c \in \mathbb{N}$ be the constant from the proof of Corollary 3.7.1. For any $n \geq ac + b$, define $f_{a,b}$ as $f_{a,b}(n) := 2^{\lfloor \frac{n}{a} - c - \frac{b}{a} \rfloor}$. Fix an arbitrary $n > \max(ac + b, 2^{a(c+2)+b})$.

By Corollary 3.6.1, there must exists a string $q_n \in \Sigma^\infty$ with $q_n = r_n\overline{0}$ for some string $r_n \in \Sigma^n$ such that $K(q_n) \geq n$.

Now, consider the string $p_n = z_{f_{a,b}(n)}\overline{0}$ from the proof of Corollary 3.7.1. By definition, $z_{f_{a,b}(n)} = 1^{f_{a,b}(n)}$. Moreover, it holds that $K(p_n) \leq \log_2(f_{a,b}(n)) + c$. But by definition, $f_{a,b}(n) = 2^{\lfloor \frac{n}{a} - c - \frac{b}{a} \rfloor} \leq 2^{\frac{n}{a} - c - \frac{b}{a}}$.

Therefore, $K(p_n) \leq \frac{n}{a} - c - \frac{b}{a} + c = \frac{n}{a} - \frac{b}{a}$. In total, we have

$$K(q_n) \geq n = a\left(\frac{n}{a} - \frac{b}{a}\right) + b \geq a \cdot K(p_n) + b. \tag{3.2}$$

At the same time, the non-zero prefix $z_{f_{a,b}(n)}$ of p_n is longer than the non-zero prefix r_n of q_n. By definition, $z_{f_{a,b}(n)}$ has length $2^{\lfloor \frac{n}{a} - \frac{b}{a} - c \rfloor} \geq 2^{\frac{n}{a} - \frac{b}{a} - c - 1}$. But as $n > \max(2^{a(c+2)+b}, 2)$, Lemma 3.11 yields $n \geq a\log_2(n) + a(c+1) + b$. By the monotonicity of the exponential function exp, this implies

$$2^n \geq 2^{a\log_2(n) + a(c+1) + b} = n^a \cdot 2^{a(c+1)+b}. \tag{3.3}$$

Latching onto this line of reasoning, the monotonicity of the square root function $\sqrt[a]{\cdot}$ now yields

$$2^{\frac{n}{a}} = \sqrt[a]{2^n} \geq \sqrt[a]{n^a \cdot 2^{a(c+1)+b}} = n \cdot 2^{c+1+\frac{b}{a}}. \tag{3.4}$$

Finally, dividing by the fixed factor $2^{c+1+\frac{b}{a}}$ culminates in

$$l\left(z_{f_{a,b}(n)}\right) = l\left(1^{f_{a,b}(n)}\right) = f_{a,b}(n) \geq 2^{\frac{n}{a} - \left(\frac{b}{a} + c + 1\right)} \geq n = l(r_n). \tag{3.5}$$

Since $z_{f_{a,b}(n)}$ consists entirely out of 1s and is longer than r_n, there must exist a $w \in \{0,1\}^*$ such that $z_{f_{a,b}(n)} = r(n) + w$, and hence $p_n = q_n + w\overline{0}$, where addition is element-wisely identified as over the field $\mathbb{Z}_r, r := |\Sigma|$.

By Equation 3.5 and the definition of w, there exist proper subsets $A, B, C \subset \mathbb{N}$ such that $q_n = \sum_{i \in A} e_i, w\overline{0} = \sum_{i \in B} e_i$, and $p_n = \sum_{i \in C} e_i$. These sets satisfy $A \cap B = \emptyset, A \cup B = C$. Since $q_n \neq p_n$, we further have $B \neq \emptyset$.

Since $\|\cdot\|$ is properly additive, repeated application of Definition 3.12 yields

$$\|p_n\| = \left\|\sum_{i \in C} e_i\right\| \overset{A \subset C}{>} \left\|\sum_{i \in A} e_i\right\| = \|q_n\|, \tag{3.6}$$

which concludes our proof. □

Lemma 3.14 (Padding Incompressible Strings)

Proof. □

Definition 3.15 (Super-logarithmic vector norms)
Let $\|\cdot\| : \Sigma^\infty \to \mathbb{R}_{\geq 0}$ be an arbitrary norm over Σ^∞. Denote by $e_1, e_2, \dots$ the unit vectors that form an orthogonal basis of Σ^∞. That is, e_i consists only out of 0s except for index i, where it carries a 1.

We say that $\|\cdot\|$ is super-logarithmic if there exist constants $c_{\|\cdot\|}, k_{\|\cdot\|} \in \mathbb{N}$ and $m_0 \in \mathbb{N}$ such that

$$\left\| \sum_{i=1}^{m} e_i \right\| \geq c_{\|\cdot\|} \left(\bigcirc_{i=1}^{k_{\|\cdot\|}} \log_2 \right)(m) \quad \text{for all } m \geq m_0. \tag{3.7}$$

Lemma 3.16 (Basis Cascade Exponential Equality)
Let $a \geq 1, b \geq 0$ be arbitrary real numbers. For any $k \in \mathbb{N}$ and $n \in \mathbb{N}$ with $n > 4 \cdot 2^k \cdot (a+b)$, it holds that

$$\left(\bigcirc_{i=1}^{n-k} \exp_2 \right)(1) > a \cdot n + b. \tag{3.8}$$

Theorem 3.17 (Unbounded Order Inconsistencies from any Super-Logarithmic Vector Norm)
Let $\|\cdot\|$ denote an arbitrary norm that is super-logarithmic as by Definition 3.15. For any real numbers $a \geq 1, b \geq 0$, there are $v, w \in \Sigma^\infty$ such that $\|v\| \geq a \cdot \|w\| + b$, but $K(v) < K(w)$.

Proof. Let $\|\cdot\|$ be an arbitrary norm that is super-logarithmic with some constants $c_{\|\cdot\|}, k_{\|\cdot\|}$. Let $a \geq 1, b \geq 0$ be arbitrary real numbers.

Denote by $c_{\mathcal{T}} \in \mathbb{N}$ the constant from the proof of Corollary 3.8.1.

For any $n \in \mathbb{N}$, Corollary 3.6.1 guarantees the existence of a string $q_n \in \Sigma^\infty$ with $q_n = r_n\overline{0}$ for some string $r_n \in \Sigma^n$ such that $K(q_n) \geq n$. Let $g : \mathbb{N} \to \Sigma^\infty$ be any selection function that maps $n \in \mathbb{N}$ to such a string q_n. In the following, we therefore denote $q_n = g(n)$.

On the other hand, define $p_n := 1^{\left(\bigcirc_{i=1}^{n} \exp_2\right)(1)}\overline{0}$ for $n \in \mathbb{N}$.

Finally, we define $f_{a,b}(n) := \left(\bigcirc_{i=1}^{k_{\|\cdot\|}} \exp_2 \right) \left(\frac{a \cdot \|q_n\| + b}{c_{\|\cdot\|}} \right)$ for any $n \in \mathbb{N}$.

Now, assume that $n > \max\left(4 \cdot 2^{k_{\|\cdot\|}} \cdot \frac{a+b}{c_{\|\cdot\|}}, 2^{4\cdot(1+c_{\mathcal{T}})} \right)$.

For the first part, we use $n > 4 \cdot 2^{k_{\|\cdot\|}} \cdot \frac{a+b}{c_{\|\cdot\|}}$ and combine Lemma 3.16 with the monotonicity of the exponential function $\exp_2$ to obtain

$$\left(\bigcirc_{i=1}^{n} \exp_2 \right)(1) = \left(\bigcirc_{i=1}^{k_{\|\cdot\|}} \exp_2 \right) \left(\left(\bigcirc_{i=1}^{n-k_{\|\cdot\|}} \exp_2 \right)(1) \right) \tag{3.9}$$

$$> \left(\bigcirc_{i=1}^{k_{\|\cdot\|}} \exp_2 \right) \left(\frac{a \cdot n + b}{c_{\|\cdot\|}} \right). \tag{3.10}$$

Since $\|\cdot\|$ is a norm, it satisfies the triangle inequality. For any $n \in \mathbb{N}$, denote by $A_n \subset \mathbb{N}$ the set such that $q_n = \sum_{i \in A_n} e_i$. As $\|e_i\| = 1$ by definition of e_i, we have

$$n \geq \sum_{i \in A_n} \|e_i\| \geq \left\| \sum_{i \in A_n} e_i \right\| = \|q_n\|. \tag{3.11}$$

By the monotonicity of the exponential function $\exp_2$, we thereby culminate in

$$\left(\bigcirc_{i=1}^{n} \exp_2\right)(1) \overset{3.9}{>} \left(\bigcirc_{i=1}^{k_{\|\cdot\|}} \exp_2\right)\left(\frac{a \cdot n + b}{c_{\|\cdot\|}}\right) \tag{3.12}$$

$$\overset{3.11}{\geq} \left(\bigcirc_{i=1}^{k_{\|\cdot\|}} \exp_2\right)\left(\frac{a \cdot \|q_n\| + b}{c_{\|\cdot\|}}\right) = f_{a,b}(n). \tag{3.13}$$

Along with Definition 3.15, our condition is therefore satisfied:

$$\|p_n\| = \left\|1^{\left(\bigcirc_{i=1}^{n} \exp_2\right)(1)}\overline{0}\right\| = \left\|\sum_{i=1}^{\left(\bigcirc_{i=1}^{n} \exp_2\right)(1)} e_i\right\| \tag{3.14}$$

$$\overset{3.15}{\geq} c_{\|\cdot\|}\left(\bigcirc_{i=1}^{k_{\|\cdot\|}} \log_2\right)\left(\left(\bigcirc_{i=1}^{n} \exp_2\right)(1)\right) \tag{3.15}$$

$$\overset{3.12}{\geq} c_{\|\cdot\|}\left(\bigcirc_{i=1}^{k_{\|\cdot\|}} \log_2\right)\left(f_{a,b}(n)\right) \tag{3.16}$$

$$= c_{\|\cdot\|}\left(\bigcirc_{i=1}^{k_{\|\cdot\|}} \log_2\right)\left(\left(\bigcirc_{i=1}^{k_{\|\cdot\|}} \exp_2\right)\left(\frac{a \cdot \|q_n\| + b}{c_{\|\cdot\|}}\right)\right) \tag{3.17}$$

$$= c_{\|\cdot\|}\left(\frac{a \cdot \|q_n\| + b}{c_{\|\cdot\|}}\right) \tag{3.18}$$

$$= a \cdot \|q_n\| + b. \tag{3.19}$$

However, since $n > 2^{4(1+c_{\mathcal{T}})}$ and by the assumptions on the compressibility of p_n and q_n, we conclude with Corollaries 3.8.1 and 3.6.1 that

$$K(p_n) \overset{3.8.1}{\leq} \log_2(n) + c_{\mathcal{T}} \overset{3.11}{<} n \overset{3.6.1}{\leq} K(q_n). \tag{3.20}$$

□

Corollary 3.17.1 (Regularization with Minkowski Norms yields no Simplicity Guarantees) *Let $p > 0$ be an arbitrary real number and $\|\cdot\|_p$ denote the corresponding Minkowski norm. For any real numbers $a \geq 1, b \geq 0$, there are both $v, w \in \Sigma^\infty$ such that $K(v) \geq a \cdot K(w) + b$, but $\|v\|_p < \|w\|_p$, and $v, w \in \Sigma^\infty$ such that $\|v\|_p \geq a \cdot \|w\|_p + b$, but $K(v) < K(w)$.*

Proof. Let $p > 0$ be an arbitrary real number.

We show that the Minkowski norm $\|\cdot\|_p$ is both properly additive (Definition 3.12) and super-logarithmic (Definition 3.15).

For the first part, let $A \subset \mathbb{N}$ be arbitrary. For any $j \in \mathbb{N}$ with $j \notin A$, the monotonicity of the root function $\sqrt[p]{\cdot}$ ensures that

$$\left\|\sum_{i \in A} e_i + e_j\right\|_p = \left(\sum_{i \in A \cup \{j\}} 1^p\right)^{\frac{1}{p}} = (|A| + 1)^{\frac{1}{p}} > |A|^{\frac{1}{p}} = \left(\sum_{i \in A} 1^p\right)^{\frac{1}{p}} = \left\|\sum_{i \in A} e_i\right\|_p. \tag{3.21}$$

For the second part, take $c = 1, k = 1$, and choose a natural number $m_0 > \max\left(\left(2^{4p}\right)^p, 2^{4p}\right)$.

Any $m \geq m_0$ in particular satisfies $m > 2^{4p}$. Therefore, Lemma A.1 guarantees that $m > p\log(m)$ for such m.

There exist a real number x such that $m = x^p$. Since $m > (2^{4p})^p$, this k satisfies $k > 2^{4p}$. For that reason, it holds for any $m \geq m_0$ that

$$\left\|\sum_{i=1}^{m} e_i\right\|_p = \sqrt[p]{m} = \sqrt[p]{k^p} = k > p\log_2(k) = p\cdot\log_2(m^{\frac{1}{p}}) = p\cdot\frac{1}{p}\log_2(m) = \log_2(m). \quad (3.22)$$

□

Corollary 3.17.2 (L0-Regularization yields no Simplicity Guarantees)
Denote by $\|\cdot\|_0$ the norm that counts non-zero components. For any real numbers $a \geq 1, b \geq 0$, there are both $v, w \in \Sigma^\infty$ such that $K(v) \geq a\cdot K(w) + b$, but $\|v\|_0 < \|w\|_0$, and $v, w \in \Sigma^\infty$ such that $\|v\|_0 \geq a\cdot\|w\|_0 + b$, but $K(v) < K(w)$.

Proof. We show that the $\|\cdot\|_0$ norm is both properly additive (Definition 3.12) and super-logarithmic (Definition 3.15).

For the first part, let $A \subset \mathbb{N}$ be arbitrary. For any $j \in \mathbb{N}$ with $j \notin A$, we have

$$\left\|\sum_{i\in A} e_i + e_j\right\|_0 = |A| + 1 > |A| = \left\|\sum_{i\in A} e_i\right\|_0 . \quad (3.23)$$

For the second part, take $c = 1, k = 1$, and choose $m_0 = 1$. For any $m \geq m_0$, it holds that

$$\left\|\sum_{i=1}^{m} e_i\right\|_0 = m > \log_2(m). \quad (3.24)$$

□

3.3 The Expressive Power of Recursion

3.4 Inductive Inference à la Solomonoff

By nature, our organism strives to minimize the energy it requires to perform a certain operation. This also applies to our brain. When trying to make sense of our world, our brain tries to do so in the most efficient way. But in terms of what kind of efficiency?

Let us illustrate this question with the example of inferring time series from few samples. Observing the sequence ... 2 _ _ _ 2..., our brains might assume the constant function $f(n) = 2$. But given ... 2 _ 4 _ _ ..., would the brain assume a linear sequence ... 2 3 4 5 ...? Or would it rather assume a constant function with one exception ... 2 2 4 2 .... And after the next sample extends the overall image to ... 2 _ 4 _ 2..., is the constant function with one exception now still the most plausible assumption? Or do we in fact deal with a symmetric piece-wise linear function ... 1 2 3 4 3 2 1 ... instead?

In general, both assumptions are reasonable, as they fit simple patterns on the observed sequences. But as more and more 2s are joining the overall image, the constant function with the exception becomes more and more plausible. While it remains a suitable candidate for the underlying pattern, alternative pattern classes become more and more complex with additional samples, and thus more and more energy-intensive. Within our inductive bias that follows some principle of simplicity, such as Ockham's Razor, the simplest algorithm becomes more and more likely to generate the observed patterns.

3.4.1 The Necessity of Sequential Inputs in Alphabetical Representation

3.4.2 Computational Complexity - The Yet Ignored Principle in Compression

The term "simplicity" might hint at the *descriptive* efficiency of an algorithm or concept. The less resources are needed to describe the algorithm, the more efficient will a machine or human be able to store this piece of information. However, this aspect does not fully capture the multi-sided shape of efficiency. Another side is the executive efficiency, that describes how many resources it needs to execute a certain algorithm. These resources include at least time, memory, but in a more general setting also communication costs between different involved units.

But given $\ldots 2\ 4\ 8\ 16 \ldots$, I do not assume a cubic polynomial, but an exponential function $f(n) = 2^n$. Although the function values are exponential in the input, the computational complexity need not be, depending on which operations the underlying architecture allows. An architecture that features bit shift operations in constant time will allow an algorithm that computes f with linear computational complexity. Moreover, its descriptive complexity is far lower than the growing complexity of the polynomial alternatives.

Chapter 4

Identifying the Simplest Consistent Algorithm

We fix a universal reference machine U and consider its induced enumeration of partial computable functions $f_1, f_2, \ldots$. Let $f : \Sigma^* \to \Sigma^*$ be an arbitrary, partial computable function over an arbitrary, but fixed, finite alphabet Σ. For any $D \subsetneq \Sigma^*$, we define

$$m_U(f \mid D) := \min_{\mathcal{S} \subset (D \times \Sigma^*)^*} \{|S| \mid \text{The smallest } f_i \text{ consistent with } \mathcal{S} \text{ satisfies } f_i \equiv f\}, \text{ and} \tag{4.1}$$

$$m_U(f) := \min_{D \subseteq \Sigma^*} m_U(f \mid D). \tag{4.2}$$

Analogously, we consider the amount of information that certainly suffices for all possible training domains and define $\mathcal{F} \subsetneq 2^{\Sigma^*}$ as the collection of subsets of Σ^* in which f is still identifiable. That is, for any partial computable function $g \not\equiv f$, $g|_D \not\equiv f|_D$ for all $D \in \mathcal{F}$. Then, we define

$$M_U(f) := \max_{D \in \mathcal{F}} m_U(f \mid D). \tag{4.3}$$

4.1 Out-Sampling Erroneous Simpler Algorithms

What qualitative and quantitative criteria must the training sample meet to ensure that the simplest algorithm that is consistent with the sample truly coincides with the true function?

4.2 Hypothesis Certification

4.3 Retrievability of the Algorithm

We avail to the idea of [8] to derive sufficient conditions when the true algorithm is retrievable.

Appendix A

Theory and Proofs

A.1 Elementary mathematics

Lemma A.1 (Log-Linear Inequality)
Let $a \geq 1$ be an arbitrary real number. For any real number $x > 2^{4a}$, $a \log_2(x) < x$.

Proof. Fix an arbitrary real number $a \geq 1$.

First of all, we show that $x < 2^x$ for all $x \in \mathbb{R}$ with $x \geq 1$ by elementary calculus. Define the real functions $g(x) = x, h(x) = 2^x$. Both g and h are continuous differentiable with derivatives $g'(x) = 1, h'(x) = \ln(2)2^x$. For any $x \geq 1$, the strict monotonicity of the exponential function yields

$$h'(x) = \ln(2)2^x \geq 2\ln(2) = \ln(4) \geq \ln(e) = 1 = g'(x). \tag{A.1}$$

But we also have $h(1) = 2 > 1 = g(1)$. Therefore, for any $x \geq 1$, it holds that

$$2^x = h(x) = h(1) + \int_1^x h'(x')dx' \overset{A.1}{>} g(1) + \int_1^x g'(x')dx' = g(1) + g(x) - g(1) = g(x). \tag{A.2}$$

In particular, the above result implies $\log_2(x) < x$ for all $x \geq 1$ by substituting $x = 2^z$ into Equation A.2. To generalize this argument to $a \log_2(n)$, we conduct an analogous argument for the real functions $g_a(x) = a \log_2(x)$ and $h(x) = x$.

By $\log_2(x) = \log_2(e) \ln(x)$ for all $x > 0$, we obtain the derivative of g_a as $g_a'(x) = \log_2(e) a \frac{1}{x}$. Since $\log_2(e) < 2$, any $x \geq 2a$ satisfies

$$g_a'(x) = \log_2(e) a \frac{1}{x} < 2a \frac{1}{2a} = 1 = h'(x). \tag{A.3}$$

Now, take $x = 2^{4a}$. Since $2^{2a} \geq 2^2 = 4$ by the assumption $a \geq 1$, it holds that

$$g_a(x) = a \log_2(x) = a \log_2(2^{4a}) = 4 \cdot a \cdot a \overset{A.2}{<} 2^{2a} \cdot 2^a \cdot 2^a = 2^{4a} = x = h(x). \tag{A.4}$$

As $2^{4a} \geq 2^{2a} \geq 2a$, for all $x > 2^{4a}$ this eventually yields

$$x = h(x) = h(2^{4a}) + \int_{2^{4a}}^{x} h'(x')dx' \tag{A.5}$$

$$\overset{A.3}{>} g_a(2^{4a}) + \int_{2^{4a}}^{x} g_a'(x')dx' \tag{A.6}$$

$$= g_a(2^{4a}) + g_a(x) - g_a(2^{4a}) = g_a(x) = a\log_2(x). \tag{A.7}$$

□

Corollary A.1.1 (Log-Linear Inequality with Additive Constant)
Let $a \geq 1, b \geq 0$ be arbitrary real numbers. For any real number $x > 2^{4(a+b)}$, $a\log_2(x)+b < x$.

Proof. Because $a \geq 1$, it is guaranteed that $x \geq 2^4 \geq 2$, and hence $\log_2(x) \geq 1$. Since $a+b \geq 1$, we employ Lemma A.1 to conclude $a\log_2(x) + b < (a+b)\log_2(x) \leq x$. □

Lemma A.2 (General Cascade Exponential Inequality)
Let a be an arbitrary real scalar with $a \geq 1$, and $k \in \mathbb{N}$. For any natural number $n \in \mathbb{N}$ with $n > 4 \cdot 2^k \cdot a$ and any real number $x \geq 2$, it holds that

$$\left(\bigcirc_{i=1}^{n-k} \exp_2\right)(x) > a \cdot n \cdot x. \tag{A.8}$$

Proof. Denote by g_2 the real function $g_2(x) = 2x$. A similarly elementary argument as in the proof of Lemma A.1 yields $\exp_2(x) = 2^x \geq 2x = g_2(x)$ for any $x \geq 2$. Repeated application of this inequality thence yields

$$\left(\bigcirc_{i=1}^{n} \exp_2\right)(x) \geq \left(\bigcirc_{i=1}^{n} g_2\right)(x) = 2^n \cdot x \quad \text{for any } n \in \mathbb{N}. \tag{A.9}$$

In particular, this states that $\left(\bigcirc_{i=1}^{n-k} \exp_2\right)(x) \geq 2^{n-k} \cdot x$ for any $n \geq k$.

Now, assume that $n > 4 \cdot 2^k \cdot a$. By Lemma A.1, it holds that $2^n > 2^k \cdot a \cdot n$. For that reason, we obtain

$$\left(\bigcirc_{i=1}^{n-k} \exp_2\right)(x) \overset{A.9}{\geq} 2^{n-k} \cdot x \tag{A.10}$$

$$\overset{A.1}{>} 2^{-k} \cdot 2^k \cdot a \cdot n \cdot x = a \cdot n \cdot x. \tag{A.11}$$

□

Corollary A.2.1 (Basis Cascade Exponential Equality)
Let $a \geq 1, b \geq 0$ be arbitrary real numbers. For any $k \in \mathbb{N}$ and $n \in \mathbb{N}$ with $n > 4 \cdot 2^k \cdot (a+b)$, it holds that

$$\left(\bigcirc_{i=1}^{n-k} \exp_2\right)(1) > a \cdot n + b. \tag{A.12}$$

Proof. Let $a \geq 1, b \geq 0$ be arbitrary real numbers. Let $n > 4 \cdot 2^k \cdot (a+b)$ be arbitrary. First of all, Equation A.2 asserts that $2^k > k$. With $a+b \geq 1$, this implies $n > 4 \cdot k \geq k+1$ for $k \geq 1$. For $k = 0$, we similarly have $n > 4 \cdot 2^k = 4 > k+1$. Therefore, we may reduce $\left(\bigcirc_{i=1}^{n-k} \exp_2\right)(1) = \left(\bigcirc_{i=1}^{n-k-1} \exp_2\right)(2)$ and conclude in the same fashion as Lemma A.2:

$$\left(\bigcirc_{i=1}^{n-k-1} \exp_2\right)(2) \overset{A.9}{\geq} 2^{-k-1} \cdot 2^n \cdot 2 \tag{A.13}$$

$$\overset{A.10}{>} 2^{-k-1} \cdot 2^k \cdot (a+b) \cdot n \cdot 2 \tag{A.14}$$

$$= (a+b) \cdot n \overset{n \geq 1}{\geq} an + b. \tag{A.15}$$

□

A.2 Out-of-distribution limitations

A.2.1 Extrapolation of ReLU MLPs with L2-Regularization

A.2.2 Prime Numbers are not learnable by IRM

A.3 Properties of Kolmogorov Complexity

A.3.1 Unbounded Kolmogorov Complexity across Reference Machines

A.3.2 No Order Preservation between Vector Norms and Kolmogorov Complexity

A.4 Invariance under Permutation

Lemma A.3 (Complexity Bound for Class of Permutations)
For an arbitrary permutation π over Σ the Turing Machine $\mathcal{T}_\pi$ that computes π as in Definition 3.9 has a Gödel number of length 10.

Proof. We briefly recall how Gödel numbers are composed on the example of $\mathcal{T}$:

- Frame Delimiters 111 at the start and the end of the Gödel number
- Inter Delimiters 11 between the encodings of each transition
- Intra Delimiters 1 between each symbol encoding inside the encoding of a transition
- Symbol Encodings of the form 0^i that encode the i-th symbol in the tape alphabet Γ, the state set Q and the direction set $\{L, R, N\}$ respectively.

Since we only have one blank symbol B, $\Gamma = \Sigma \cup \{B\}$. A Turing Machine with Q states among which exactly one identifies the final state q_1 has a transition function δ with a definition range of cardinality $|\Gamma| \cdot (|Q| - 1) = (|\Sigma| + 1) \cdot (|Q| - 1)$. Therefore, we require $(|\Sigma| + 1) \cdot (|Q| - 1) - 1$ Inter Delimiters. Moreover, the encoding of each functional tuple $(q, a, q', a', d) \in Q \times \Gamma \times Q \times \Gamma \times \{L, R, N\}$ requires four intra delimiters. Since $\mathcal{T}_\pi$ computes a permutation, every symbol must appear equally often in the encoding. In the q_0 state where $\mathcal{T}_\pi$ replaces each symbol b by $\pi(b)$, every symbol must occur exactly once in the second a forth position of the tuple (q_0, a, q', a', d). As soon as $\mathcal{T}_\pi$ reaches the blank symbol B, it writes B again and shifts into state q_2.

The same holds in state q_2 where $\mathcal{T}_\pi$ merely traverses back over the input to the first symbol. Each symbol b is replaced with itself, and therefore occurs exactly once in the second and forth position of tuples of the form (q_2, a, q', a', d), too.

As the overview in Table 3.1 illustrates, state changes only take place when $\mathcal{T}_\pi$ encounters the blank symbol at the right or left end of the word. For tuples of the form (q_0, a, q', a', d), we therefore only have one tuple with $q' \neq q_0$, namely (q_0, B, q_2, B, L).

Start and End Delimiter 111	$2 \cdot 3 = 6$
Inter Transition Delimiter 11	$((\lvert Q\rvert - 1) \cdot (\lvert\Sigma\rvert + 1) - 1) \cdot 2$
Intra Transition Delimiter 1	$4 \cdot (\lvert Q\rvert - 1) \cdot (\lvert\Sigma\rvert + 1)$
From States	$(\lvert\Sigma\rvert + 1) \cdot 1$ (q_0 tuples) $(\lvert\Sigma\rvert + 1) \cdot 3$ (q_2 tuples)
To States	$\lvert\Sigma\rvert \cdot 1 + 3$ (q_0 tuples) $\lvert\Sigma\rvert \cdot 3 + 2$ (q_2 tuples)
Symbols	$2 \cdot 2 \cdot \sum_{i=1}^{\lvert\Sigma\rvert+1} i = 2(\lvert\Sigma\rvert + 1)(\lvert\Sigma\rvert + 2)$
Directions	$\leq (\lvert\Sigma\rvert + 1) \cdot (2 + 3)$
Total	$2\lvert\Sigma\rvert^2 + 31\lvert\Sigma\rvert + 34$

Table A.1: Gödel number of $\mathcal{T}_\pi$ for any permutation π over a finite alphabet Σ. For the sake of clarity, $|Q|$ was specified in the terms although $|Q| = 3$ is constant across all Σ. To preserve generality, the worst case where the directions L and R possess the worst encoding lengths 2 and 3 are assumed.

Similarly, the only tuple of the form (q_2, a, q', a', d) with $q' \neq q_2$ is (q_2, B, q_1, B, R).

The direction symbols L and R appear equally often in the tuples. When defining the universal Turing Machine in Definition 3.3, we did not specify the specific ordering of the direction indicator set $\{L, R, N\}$. As this only has a tiny impact on the final constant, we can generously assume that L and R are associated with the longest encodings $\{0^2, 0^3\}$.

An overview for the overall number of bits of $\mathcal{T}_\pi$'s Gödel number is depicted in Table A.1. With $|Q| = 3$, we accordingly conclude that the Gödel number of $\mathcal{T}_\pi$ has a length bounded by

$$a \cdot |\Sigma|^2 + b \cdot |\Sigma| + c, \text{ where} \tag{A.16}$$

$$a = 2, \tag{A.17}$$

$$b = 4 + 8 + 1 + 3 + 1 + 3 + 6 + 5 = 31, \tag{A.18}$$

$$c = 6 + 2 + 8 + 1 + 3 + 3 + 2 + 4 + 5 = 34. \tag{A.19}$$

This length is invariant of the specific permutation π but holds equally across the entire class of permutations.

Recall that the universal Turing Machine U encodes a program p not by its explicit Gödel number but by the index i it possesses in the natural enumeration of valid Gödel numbers. Since a large majority of the strings in $\{0, 1\}^*$ do not identify valid Gödel numbers, the encoding of this index is smaller than the encoding of the Gödel number itself. Therefore, the above bound for the Gödel number enc($\mathcal{T}_\pi$) also serves as a more loose bound for the Kolmogorov complexity $K(\mathcal{T}_\pi)$ of the class of permutations over Σ. □

Lemma A.4 (Optimality of $\mathcal{T}_\pi$)
For any permutation π over Σ besides the identity function, there exists no program p with a shorter Gödel number

Proof. Let π be an arbitrary permutation over Σ that is not equivalent to the identity function $I : a \mapsto a$. Let $\mathcal{T}$ be a Turing Machine that computes π. In the following, we denote by B the default blank symbol on the tape. Besides that, we fix q_0 as the initial state and q_1 as the final state just as in the natural encoding.

$\mathcal{T}$ must encounter each symbol of the input at least once. Since the Turing Machine's tape head starts over the leftmost symbol of the input, it must necessarily encounter the blank symbol B right next to the input. Otherwise, there would be a symbol in the input it never traverses over, and hence never permutes. This would violate the assumption that $\mathcal{T}$ computes π for all possible inputs $x \in \Sigma^*$.

When a Turing Machine halts, the output is interpreted as to start from the symbol the current tape head is positioned over. Therefore, the Turing Machine must move its head back to the start of the permuted sequence. By the same argument as above, it must encounter the blank symbol right left to the sequence.

Now assume $\mathcal{T}$ had only one non-final state. Denote this non-final state by q_0. In both situations, where $\mathcal{T}$ encounters the right end and left end of the sequence, the input to its transition function δ is necessarily (q_0, B). For that reason, $\mathcal{T}$ must execute the same operation $\delta(q_0, B) = (q, a, d)$ in both situations.

When $\mathcal{T}$ encounters the blank symbol right next to the sequence, it may not halt, because it still has to move back to the start of the sequence. For that reason, $q = q_0$. Moreover, $d \neq N, R$, since $\mathcal{T}$ would never halt and compute perpetually when it reaches the the blank symbol next to the input sequence. But at the same time, $d \neq L$, because $\mathcal{T}$ would drift off for the same reason when it reaches the blank symbol left to the input sequence. This leads to a contradiction. $\lightning$

Therefore, $\mathcal{T}$ must have at least two non-final states. No matter how $\mathcal{T}$ operates, it must traverse at least once from left to right over the sequence and at least once back again. There must be two different states representing these different directions by the above argument. The most efficient encodings for these states are 0^1 and 0^3 for q_0 and q_2 respectively. This means that there must occur at least as many delimiters 111,11, and 1 as in the Gödel number of $\mathcal{T}_\pi$. Moreover, there must be at least $(|\Sigma| + 1) \cdot (1 + 3)$ 0s to encode the first element in the transition tuples from states q_0, q_2.

In the end, each symbol a in the input sequence must have been overwritten at least once by a symbol $b(a)$, which is read again on the way back to the start and eventually surely overwritten by $\pi(a)$. This mapping b must be a bijective function too, otherwise the image of b would not encompass all symbols in Σ and there would be some input strings x for which $\mathcal{T}$ does not correctly compute the permutation, again contradicting our assumption.

Since b and π are both bijective, each symbol encoding must occur equally often in the transitions of at least two states by the same argument as in the proof of Lemma A.3. Henceforth, we require at least $\underbrace{2}_{q_0, q_2} \cdot \underbrace{2}_{(q, \cdot, q', \cdot, d)} \cdot \underbrace{\sum_{i=1}^{|\Sigma|+1} i}_{\text{equally often}}$ 0s to encode the symbols at the second and forth positions of the transition tuples.

Similarly, the directions L and R must occur equally often in the image of δ restricted on input symbols $\Sigma \cup \{B\}$. Consequently, we require at least $(|\Sigma| + 1) \cdot (2 + 3)$ 0s to encode these directions, assuming the worst-case encoding of L and R as in the proof of Lemma

A.3.

Finally, we inevitably require some transition from q_0 to another non-final state that represents the directional change. Eventually, a transition to q_1 is required in the very end. As the most efficient encodings for the two states that traverse over the input are achieved by choosing q_0, q_2, the encodings for the third symbol in the tuples necessarily requires $\underbrace{|\Sigma| \cdot 1}_{0^1 \text{ for } q_0} + \underbrace{|\Sigma| \cdot 3}_{0^3 \text{ for } q_2} + \underbrace{3+2}_{\text{transitions to } q_2, q_1}$ 0s.

In total, the encoding of $\mathcal{T}$ hence can not be shorter than $\mathcal{T}_\pi$, as these lower bounds are exactly the encodings listed in Table A.1 for $\mathcal{T}_\pi$. This concludes the proof. □

A.5 Information Thresholds for Learnability

A.5.1 Abysmal Minimal Sample Count to Kolmogorov Complexity Ratio

We are interested if there exists a function m that - given the Kolmogorov complexity of a function f - serves a lower bound on the required sample count to identify f in the learning by enumeration framework. As it turns out, there is no effective lower bound. In other words, the sample count might become as small as one for arbitrarily high Kolmogorov complexities.

This demonstrates that it should not be the sample count we seek to relate the Kolmogorov complexity to, but instead quantify the training sample itself by its Kolmogorov complexity, too.

Proof.

Since Σ^* is countably infinite, consider an arbitrary order of Σ^* as $x_1, x_2 \ldots$. Fix an arbitrary definition range $D \subseteq \Sigma^*, D \neq \emptyset$, and an arbitrary $c \in D$. For any x_j, let f_{i_j} be the simplest function consistent with $\mathcal{S}_j := \{(c, x_j)\}$. Then, f_{i_j} is learnable from the Kolmogorov enumeration by a sample of size 1.

Obviously, $f_{i_j} \not\equiv f_{i_k}$ for any $j, k \in \mathbb{N}$ with $j \neq k$, since they disagree for input c.

Therefore, there are infinitely many functions over Σ^* that are learnable from the Kolmogorov enumeration by a sample of size 1. However, for each $n \in \mathbb{N}$, there are only finitely many objects with Kolmogorov complexity n.

A.5.2 Kolmogorov Complexity Upper Bound for K_U

A.5.3 Unbounded K_U across U for fixed finite D

A.5.4 Unbounded K_U across U for fixed countably infinite D

A.5.5 Lower Bound for K_U

References

[1] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, “Understanding deep learning (still) requires rethinking generalization,” *Communications of the ACM*, vol. 64, no. 3, pp. 107–115, 2021.

[2] B. M. Lake, T. D. Ullman, J. B. Tenenbaum, and S. J. Gershman, “Building machines that learn and think like people,” *Behavioral and brain sciences*, vol. 40, e253, 2017.

[3] M. Li, P. Vitányi, *et al.*, *An introduction to Kolmogorov complexity and its applications.* Springer, 2008, vol. 3.

[4] J. Peters, P. Bühlmann, and N. Meinshausen, “Causal inference by using invariant prediction: Identification and confidence intervals,” *Journal of the Royal Statistical Society Series B: Statistical Methodology*, vol. 78, no. 5, pp. 947–1012, 2016.

[5] N. Pfister, P. Bühlmann, and J. Peters, “Invariant causal prediction for sequential data,” *Journal of the American Statistical Association*, vol. 114, no. 527, pp. 1264–1276, 2019.

[6] M. Arjovsky, L. Bottou, I. Gulrajani, and D. Lopez-Paz, “Invariant risk minimization,” *arXiv preprint arXiv:1907.02893*, 2019.

[7] K. Ahuja, E. Caballero, D. Zhang, J.-C. Gagnon-Audet, Y. Bengio, I. Mitliagkas, and I. Rish, “Invariance principle meets information bottleneck for out-of-distribution generalization,” *Advances in Neural Information Processing Systems*, vol. 34, pp. 3438–3450, 2021.

[8] J. Richens and T. Everitt, “Robust agents learn causal world models,” *arXiv preprint arXiv:2402.10877*, 2024.

[9] K. Gödel, “Über formal unentscheidbare sätze der principia mathematica und verwandter systeme i,” *Monatshefte für mathematik und physik*, vol. 38, pp. 173–198, 1931.