CS3236 Project: Kolmogorov Complexity

Hu Hanyang

2023-04-17

1 Introduction

How would you measure the amount of information in a file? The number of bits used to store that file is not a very good measure. For example, an image file might have 640×640 raw pixels, but we can always take a compression method of digital images (PNG, JPEG, etc.) to store it with fewer bits. Moreover, we may be lucky enough to devise a very short program that generates the entire image, hence further compressing the image. It seems as if there is always a better way to compress the image and store it in fewer bits.

Surprisingly, in the 1960s, great mathematicians such as Kolmogorov independently discovered that there exists an optimal compression method that defines an invariant measure of information content for a binary string (e.g. a file stored in our computer). Here comes the notion of Kolmogorov complexity: the smallest number of bits we need to compress a binary string using a fixed optimal compression method.

The optimal compression method relates to the notion of universal Turing machines introduced by Alan Turing. In this report, we shall define Kolmogorov complexity with a fixed universal Turing machine and explore its connection with Shannon's information theory.

2 The Concept of Kolmogorov Complexity

Rather than delving into the intricacies of formalizing computers and algorithms, the primary objective of this section is to provide an intuitive and practical understanding of the concept of Kolmogorov complexity.

2.1 Algorithmic Complexity and the Invariance Theorem

We will take computers as a specific kind of partial function $\mathcal{C} : \{0,1\}^* \rightharpoonup \{0,1\}^* \cup \{0,1\}^\infty$.[1] For readers who may not be familiar with the concept of partial functions:

Definition 2.1.1. A partial function $f : X \rightharpoonup Y$ from set X to set Y is a function from a subset $S \subseteq X$ to Y. The subset S is called the natural domain of f. If $x \in X \setminus S$, then $f(x)$ is said to be undefined.

In the case of computers, the input is typically a finite-length binary code that serves as a program, which the computer executes to produce a finite- or infinite-length[2] output.

Definition 2.1.2. The complexity $K_\mathcal{C}(x)$ of a string $x \in \{0,1\}^* \cup \{0,1\}^\infty$ with respect to a computer $\mathcal{C}$ is

$$K_\mathcal{C}(x) := \min_{p \,:\, \mathcal{C}(p)=x} l(p).$$

Remark. If there does not exist $p \in \{0,1\}^*$ such that $\mathcal{C}(p) = x$, we may assume that $K_\mathcal{C}(x) = +\infty$.

Computer scientists Ray Solomonoff and Andrey Kolmogorov independently discovered a fundamental idea that the complexity of strings is an inherent property, independent of computers and dispenses with probability distributions. More specifically, among computers that decode strings from their descriptions (codes), there exists an optimal one (up to $O(1)$ additive term).

Theorem 2.1.3. (The invariance theorem) There exists an optimal computer $\mathcal{U}$, given any computer $\mathcal{C}$, there exists a constant c such that

$$K_\mathcal{U}(x) \le K_\mathcal{C}(x) + c$$

for all $x \in \{0,1\}^* \cup \{0,1\}^\infty$.

[1]This is not a proper definition of computers, as it does not encompass all the necessary characteristics of computers to ensure the existence of a "universal computer" (at least, we need the collection of all computers to be countable). For a more rigorous formalization, readers may turn to the concept of a Turing machine, which is an abstract device capable of executing any algorithms by manipulating symbols on a tape according to a set of rules.

Alternatively, analogous to Shannon's entropy which could be characterized by a list of properties, Kolmogorov complexity can also be axiomatized by several axioms that capture its essential properties without associating with the formal definition of computers. See sections 8, 9, and 10 on Shen's lecture note *Algorithmic Information Theory and Kolmogorov Complexity.*

[2]A Turing machine is capable of generating infinite-length strings if they are produced by an infinite loop in the program.

Remark. The existence of such an optimal algorithm is guaranteed by the existence of universal computers (the UTM theorem), i.e. there exists a universal computer $\mathcal{U}$, given any computer $\mathcal{C}$, there exists a prefix $e \in \{0,1\}^*$ such that $\mathcal{U}(e+p) = \mathcal{C}(p)$ for all p in the natural domain of $\mathcal{C}$.[3] Taking $c = l(e)$, we have

$$K_{\mathcal{U}}(x) = \min_{p\,:\,\mathcal{U}(p)=x} l(p) \leq \min_{p\,:\,\mathcal{U}(e+p)=x} l(e+p) = \min_{p\,:\,\mathcal{C}(p)=x} l(p) + l(e) = K_{\mathcal{C}}(x) + c$$

By symmetry, given any two universal computers $\mathcal{U}_1, \mathcal{U}_2$, there exists a constant c such that

$$|K_{\mathcal{U}_1}(x) - K_{\mathcal{U}_2}(x)| \leq c$$

for all $x \in \{0,1\}^* \cup \{0,1\}^\infty$.

Consequently, it is justifiable to define the Kolmogorov complexity as the complexity of strings with respect to a fixed unspecified universal computer $\mathcal{U}$, and the subscript $\mathcal{U}$ can be dropped from its notation.[4]

Definition 2.1.4. The Kolmogorov complexity $K(x)$ of a string $x \in \{0,1\}^* \cup \{0,1\}^\infty$ is defined by

$$K(x) := K_{\mathcal{U}}(x).$$

Analogous to Shannon's entropy, we may define conditional (resp. joint) Kolmogorov complexity.

Definition 2.1.5. The conditional Kolmogorov complexity of two strings $x \in \{0,1\}^* \cup \{0,1\}^\infty$ and $y \in \{0,1\}^*$ is defined by

$$K(x|y) := \min_{p\,:\,\mathcal{U}(p,y)=x} l(p)$$

where y is the auxiliary input.

Definition 2.1.6. The joint Kolmogorov complexity of two strings $x, y \in \{0,1\}^*$ is defined by

$$K(x,y) := K([x,y])$$

where the pairing function $[\cdot,\cdot] : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$ is a fixed bijection that is computable[5] and has a computable inverse.

Remark. Given two pairing functions f_1, f_2 that satisfy the requirements, there exists a constant c such that $|K(f_1(x,y)) - K(f_2(x,y))| \leq c$ for all $x, y \in \{0,1\}^*$ because $f_2 \circ f_1^{-1}$ and $f_1 \circ f_2^{-1}$ are computable.

By similar reasoning, the joint Kolmogorov complexity is "symmetric" in the sense that there exists a constant c such that $|K(x,y) - K(y,x)| \leq c$.

Moreover, there does exist a bijection $[\cdot,\cdot] : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$ that is computable and has computable inverse. For instance, define $B : \{0,1\}^* \to \mathbb{Z}_{\geq 0}$ by $B(b_1, b_2, \ldots, b_n) = (\overline{b_1 b_2 \cdots b_n})_2$ and $[\cdot,\cdot] : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$ by $[x,y] = B^{-1}((B(x)+B(y))(B(x)+B(y)+1)/2 + B(y))$. One may check that this pairing function is bijective, computable and has a computable inverse.

Consequently, the definition of joint Kolmogorov complexity makes sense.

In Section 2.2, we will discover that some fundamental properties of Kolmogorov complexity align with those of Shannon's entropy. However, we will see that such alignment comes with a logarithmic error term in certain cases. Consequently, a refined version of Kolmogorov complexity has been proposed, which "has precisely the formal properties of the entropy concept of information theory"[6].

[3] Unless stated otherwise, the binary operation $+ : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$ refers to binary string concatenation.

[4] Unless stated otherwise, from now on, $\mathcal{U}$ refers to a fixed universal computer.

[5] A function f is computable if there exists a (universal) computer $\mathcal{U}$ and a string $p \in \{0,1\}^*$ such that $\mathcal{U}(p+x) = f(x)$ for all $x \in \{0,1\}^*$, i.e. f can be computed by a computer. Partial computable functions can be defined similarly. Notice that not all computable function has (partial) computable inverse.

[6] In Gregory J. Chaitin's paper *A Theory of Program Size Formally Identical to Information Theory.*

The differing feature of this refined version, namely self-delimiting Kolmogorov complexity, is that it assumes programs to be self-delimiting, i.e. the natural domain of a computer should always be prefix-free. Technically, this allows us to be more aligned with information-theoretic results (e.g. Kraft inequality). Intuitively, the prefix-free constraint reflects the convention that a program is validly executed on a computer if the process halts and the program is read entirely.[7] It is worth noting that the two versions of Kolmogorov complexity only differ up to a logarithmic term.

In this report, we will majorly rely on the Kolmogorov complexity $K(\cdot)$ in Definition 2.1.4. When it comes to self-delimiting Kolmogorov complexity, we will use the notation $K'(\cdot)$.

2.2 Fundamental Properties of Kolmogorov Complexity

Theorem 2.2.1. (Upper bound on Kolmogorov complexity) For all $x \in \{0,1\}^* \cup \{0,1\}^\infty$, we have constants c s.t. $K(x) \leq l(x) + c$.

Proof. We may devise a program that prints the $l(x)-$bit sequence $(x_1, x_2, \dots, x_{l(x)})$ one by one, which takes length of $l(x) + O(1)$. Hence $K(x) \leq l(x) + c$ for some constant c. $\square$

Remark. For self-delimiting Kolmogorov complexity, unless $l(x)$ is conditioned, we have to encode $l(x)$ and make sure that the program splits the encoding of $l(x)$ and the program $p_x \in \{0,1\}^*$ (of length $l(x) + O(1)$) that prints x (in order to let the program know when to terminate).

Firstly, encode $l(x)$ as a binary number, then double each digit and attach a string "01" at the end. Concatenate this string with p_x and devise a program that locates the first "01" pattern which separates the encoding of $l(x)$ and p_x, then we can print the $l(x)-$bit sequence by p_x. The entire program takes a length of $l(x) + 2\log l(x) + c$ for some constant c.

Moreover, the encoding method that encodes $l(x)$ can be improved recursively. For instance, we can encode $l(l(x))$ and obtain an improved upper bound of $l(x) + \log l(x) + 2\log\log l(x) + O(1)$. Repeat the process until the last positive term, the sum of the iterated logarithms is denoted by $\log^* n$. Consequently, the upper bound of self-delimiting Kolmogorov complexity can be improved to $l(x) + \log^* l(x) + O(1)$.[8]

Example. For $x = (1,0,1,1,0)$, $l(x)$ can be encoded by $(1,1,0,0,1,1,0,1)$ since $l(x) = (101)_2$.

Theorem 2.2.2. (Lower bound on Kolmogorov complexity) The number of strings x with complexity $K(x) < k$ satisfies

$$|\{x \in \{0,1\}^* \mid K(x) < k\}| < 2^k.$$

Proof. For any string x s.t. $K(x) < k$, there exists a unique program p s.t. $\mathcal{U}(p) = x$ and $l(p) < k$, hence

$$\begin{aligned} |\{x \in \{0,1\}^* \mid K(x) < k\}| &\leq |\{p \in \{0,1\}^* \mid \mathcal{U}(p) = x \text{ and } l(p) < k\}| \\ &= \sum_{i=0}^{k-1} |\{p \in \{0,1\}^* \mid \mathcal{U}(p) = x \text{ and } l(p) = i\}| \\ &\leq \sum_{i=0}^{k-1} 2^i = 2^k - 1 < 2^k. \end{aligned}$$

$\square$

[7]Given programs $p, p' \in 0,1^*$, if p is a prefix of p' and p is validly executed on a computer $\mathcal{C}$, it will terminate once it has finished reading p. Thus, it cannot continue to read the remaining part of p', and therefore p' cannot be a valid program for $\mathcal{C}$.

[8]This encoding method is from Cover's proof of Theorem 14.2.3 in *Elements of Information Theory*.

Remark. Informally speaking, "typical" binary string of length n has complexity close to n. Assume that each bit of x is sampled from $Bernoulli(\frac{1}{2})$ independently, then we have

$$\begin{aligned}
\mathbb{P}[n-c \leq K(x) \leq n+c \,|\, l(x)=n] &= \mathbb{P}[K(x) \geq n-c \,|\, l(x)=n] && \text{(Theorem 2.2.1)} \\
&= 1-\mathbb{P}[K(x) < n-c \,|\, l(x)=n] \\
&= 1-\frac{|\{x \in \{0,1\}^n \,|\, K(x) < n-c\}|}{|\{0,1\}^n|} \\
&> 1-\frac{2^{n-c}}{2^n} = 1-2^{-c}
\end{aligned}$$

where c satisfies $c \geq K(x)-l(x)$ for all x (from Theorem 2.2.1). [9]

Theorem 2.2.3. (Algorithmic transformation does not increase complexity) Given any computable function f, there exists a constant c such that

$$K(f(x)) \leq K(x)+c$$

for all binary string x.

Proof. Since f is computable, $f \circ \mathcal{U}$ is computable and there exists a binary string e and a computer $\mathcal{C}$ such that $\mathcal{C}(e+y) = f(\mathcal{U}(y))$ for all binary string y. Hence $K_{\mathcal{C}}(f(x)) \leq K_{\mathcal{U}}(x)+c'$ for some constant c'. [10] By Theorem 2.1.3, we have $K_{\mathcal{U}}(f(x)) \leq K_{\mathcal{C}}(f(x))+c \leq K_{\mathcal{U}}(x)+c+c'$. Hence the inequality. □

Theorem 2.2.4. (Conditioning does not increase complexity) There exists a constant c such that

$$K(x|y) \leq K(x)+c$$

for any binary strings x, y.

Proof. There exists a universal computer $\mathcal{C}$ such that $\mathcal{C}(p) = \mathcal{U}(p,y)$ which simply ignores the condition y. The inequality follows immediately from Theorem 2.1.3. □

Remark. For any fixed condition y, there exists a constant c' such that $K(x) \leq K(x|y)+c'$ for all binary string x. Intuitively, a fixed auxiliary input does not reduce complexity in general.

Theorem 2.2.5. (Sub-additivity of Kolmogorov complexity) There exists a constant C such that

$$K(x,y) \leq K(x)+K(y)+C\log(\min(K(x),K(y)))$$

for all binary strings x, y.

Proof. Let $p_x := \underset{p\,;\,\mathcal{U}(p)=x}{\arg\min}\ l(x)$ and $p_y := \underset{p\,:\,\mathcal{U}(p)=y}{\arg\min}\ l(p)$. WLOG, suppose $K(x) \leq K(y)$, i.e. $l(p_x) \leq l(p_y)$. We will use the same encoding method in the proof of Theorem 2.2.1 to encode $l(p_x)$ (which takes $2\log l(p_x)+2$ bits), then concatenate with the binary string p_x+p_y. We could devise a program that splits the encoding of $l(p_x)$ and p_x+p_y, then utilize the information of $l(p_x)$ to split p_x and p_y from p_x+p_y. Subsequently, we can calculate $[x,y] = [\mathcal{U}(p_x),\mathcal{U}(p_y)]$. In conclusion, this program input of $\mathcal{U}$ to compute $[x,y]$ is of length $2\log l(p_x)+2+l(p_x)+l(p_y)+c$ where c is a constant (the length of the program that splits p_x and p_y to calculate $[\mathcal{U}(p_x),\mathcal{U}(y)]$). Consequently, we have shown Theorem 2.2.5. □

Remark. According to Li and Vitányi (1990), Theorem 2.2.5 cannot be generally improved. However, if we assume that programs are self-delimiting, then it would be unnecessary to encode $l(p_x)$, hence we may have $K'(x,y) \leq K'(x)+K'(y)+c$ for a constant c.

[9] For self-delimiting Kolmogorov complexity, replace $K(x)$ with $K'(x|l(x))$.

[10] Notice that for $p' = \underset{p\,:\,\mathcal{U}(p)=x}{\arg\min}\ l(p)$, we have $l(p') = K_{\mathcal{U}}(x)$ and $\mathcal{C}(e+p') = f(x)$, hence $K_{\mathcal{C}}(f(x)) \leq l(e+p') = K_{\mathcal{U}}(x)+l(e)$.

Lemma 2.2.6. There exists a constant c s.t. $K(x) \leq K(x,y) + c$ for all binary strings x, y.

Proof. One inefficient way to compute x is to compute $[x,y]$ first (with a program of length $K(x,y)$), then compute the inverse and obtain the value of x (with a program of constant length). Hence the inequality. $\square$

Theorem 2.2.7. (Chain rule of Kolmogorov complexity) For binary strings x, y, we have

$$K(x,y) - (K(x) + K(y|x)) = O(\log(K(x,y))).$$

Proof. Obviously, we have $K(x,y) \leq K(x) + K(y|x) + C\log K(x) \leq K(x) + K(y|x) + C\log K(x,y)$ for some constant C because we can devise a program that computes $[x,y]$ by computing x with a program of length $K(x)$, then compute y given x with a program of length $K(y|x)$ and encode $K(x)$ with $2\log K(x) + 2$ bits to tell where to separate the two programs, then compute $[x,y]$ which takes a program of constant length.

On the other hand, we need to have $K(x) + K(y|x) \leq K(x,y) + C' \log(K(x,y))$ for some constant C'.

Define $a := K(x,y)$, $A := \{(p,q) \,|\, K(p,q) \leq a\}$, $A_p := \{q \,|\, (p,q) \in A\}$ and $b_p := \lfloor \log(|A_p|) \rfloor$ for each string p. Specifically, let $b = b_x$. Notice that A is computably enumerable[11] given a (by the algorithm that runs all programs p s.t. $l(p) \leq a$, of which there are at most 2^a, in parallel). Hence we may also enumerate A_x by enumerating A first (only output the second entry of elements in the list of A when the first entry is x). In this sense, given auxiliary input x, we may describe y by concatenating the encoding[12] of its ordinal number in A_x (which takes $b + O(\log b)$ bits) and the program that enumerates A_x (which takes $O(\log a)$ bits because we need to specify a).

In total, the program that computes y conditioned on x has a length of $b + O(\log a)$, that is to say $K(y|x) \leq b + C_1^{'} \log a$ where $C_1^{'}$ is a constant.

Moreover, let $X := \{x' \,|\, |A_{x'}| \geq 2^b\}$, then $x \in X$ (by definition), $|X| \leq 2^{a-b}$ (otherwise, $\sum_{x'} |A_{x'}| \geq \sum_{x' \in X} |A_{x'}| > 2^{a-b}2^b = 2^a \geq |A|$, a contradiction) and X can also be enumerated by enumerating A first (in this case, we have to specify a and b).

Therefore, we could compute x by enumerating X (which takes $O(\log a) + O(\log b)$ bits) and find x by its encoded ordinal number in X (which takes at most $a - b + O(\log(a-b))$ bits), hence $K(x) \leq a - b + C_2^{'} \log a$ where $C_2^{'}$ is a constant.

Consequently, we have shown that $K(x) + K(y|x) \leq (a - b + C_2^{'} \log a)) + (b + C_1^{'} \log a)) = a + (C_1^{'} + C_2^{'}) \log a = K(x,y) + C^{'} \log(K(x,y))$ where $C^{'} = C_1^{'} + C_2^{'}$. Hence Theorem 2.2.7 is proved.[13] $\square$

Remark. Theorem 2.2.7 allows us to establish the notion of algorithmic mutual information.

Definition 2.2.8. The algorithmic mutual information of two binary strings x, y is defined by

$$I_K(x;y) := K(x) - K(x|y).$$

Remark. Notice that the error term of "symmetricity" is $I_K(x;y) - I_K(y;x) = O(\log(K(x,y))$. When it comes to self-delimiting Kolmogorov complexity, the chain rule would be $K'(x,y) = K'(x) + K'(y|x) + O(1)$, hence $I_{K'}(x;y) - I_{K'}(y;x) = O(1)$.[14]

[11]A set S is called computably enumerable if there is an algorithm such that the set of input numbers for which the algorithm halts is exactly S, or equivalently, there is an algorithm that outputs all members of S in a list.

[12]In this proof, we will use the encoding method of Cover in proving the self-delimiting version of Theorem 2.2.1 to encode the ordinal numbers of elements in an enumerable set.

[13]This proof is adapted from Alexander Shen's lecture note and the corresponding Wikipedia page on the chain rule for Kolmogorov complexity, available at `https://en.wikipedia.org/wiki/Chain_rule_for_Kolmogorov_complexity`.

[14]The proof would be different from how we prove Theorem 2.2.8 in this report. Interested readers may refer to Chaitin's paper *A Theory of Program Size Formally Identical to Information Theory.*

3 Connections with Shannon's Entropy

In this section, we will explore the theoretical connection between Kolmogorov complexity and Shannon's entropy, which will shed light on the apparent "coincidence" we observed in the previous section.

Lemma 3.1. The Kolmogorov complexity of a binary string $x = (x_1, x_2, \dots, x_n)$ is bounded by

$$K(x) \le nH_2\left(\frac{1}{n}\sum_{i=1}^{n} x_i\right) + C\log n$$

for some constant C.

Proof. Let $k = \sum_{i=1}^{n} x_i$. Devise program p as follows: given inputs k and i, generate, in shortlex order[15], all sequences with k ones until the ith sequence, then print the ith sequence. To specify x, we only need to encode program p (of constant length) and the inputs k, i so that x is the ith sequence with k ones.

Notice that $k \le n$ and $i \le \binom{k}{k} + \binom{k+1}{k} + \cdots + \binom{n}{k} = \binom{n+1}{k+1} = \frac{n+1}{k+1}\binom{n}{k}$ (by the Hockey-stick identity).

By a corollary of Stirling's approximation[16], we have $\binom{n}{k} \le \sqrt{\frac{n}{\pi k(n-k)}}\, 2^{nH_2(\frac{k}{n})}$, hence there exists a constant C' such that

$$\log i \le \log\left[\frac{n+1}{k+1}\binom{n}{k}\right] \le nH_2\left(\frac{k}{n}\right) + C'\log n$$

Consequently, we may check that there exists a constant C s.t. $K(x) \le nH_2(\frac{k}{n}) + C\log n$, by choosing a proper encoding of p, k, i.[17] $\square$

Lemma 3.2. Let $X_1, X_2, \dots, X_n$ be drawn i.i.d. $\sim$ *Bernoulli*(θ). Fix an arbitrary $\epsilon > 0$, we have

$$\mathbb{P}\left[H_2(\theta) - \frac{1}{n}K(X_1X_2\cdots X_n) \ge \epsilon\right] \to 0$$

Proof. By Theorem 2.2.2, we have

$$\mathbb{P}[K(X_1X_2\cdots X_n) < n(H_2(\theta) - \epsilon)] < \frac{2^{n(H_2(\theta)-c)}}{2^n} = 2^{n(H_2(\theta)-\epsilon-1)}$$

Hence

$$\mathbb{P}\left[H_2(\theta) - \frac{1}{n}K(X_1X_2\cdots X_n) > \epsilon\right] < 2^{n(H_2(\theta)-\epsilon-1)}$$

Notice that $H_2(\theta) \le 1 < 1 + \epsilon$ for all $\theta \in [0,1]$, hence Lemma 3.2 follows immediately. $\square$

Lemma 3.3. Let $X_1, X_2, \dots, X_n$ be drawn i.i.d. $\sim$ *Bernoulli*(θ). Then

$$\frac{1}{n}K(X_1X_2\cdots X_n) \to H_2(\theta) \quad \text{in probability.}$$

Proof. Let $\overline{X_n} = \frac{1}{n}\sum_{i=1}^{n} X_i$, by Lemma 3.1, we have

$$\frac{1}{n}K(X_1X_2\cdots X_n) - H_2(\overline{X_n}) \le C\frac{\log n}{n}$$

[15]Shortlex ordering sorts sequences primarily by length, with shorter sequences coming first, and then by lexicographical order when sequences have the same length.

[16]See Cover's *Elements of Information Theory*, Lemma 17.5.1. Our proof for Lemma 3.1 is modified from his proof of the corresponding self-delimiting version (Theorem 14.2.5).

[17]More specifically, by the encoding method Cover used in proving the self-delimiting version of Theorem 2.2.1, we could encode p with $2l(p)+2 = O(1)$ bits; encode k with $2l(k)+2 = O(\log n)$ bits; and encode i with $[2l(l(i))+2]+l(i) = l(i)+O(\log n)$ bits (encode $l(i)$ first to specify that the following $l(i)$ bits represent i).

By the weak law of large numbers, we have $\overline{X_n} \to \theta$ in probability. Hence

$$\mathbb{P}\left[\frac{1}{n}K(X_1X_2\cdots X_n) - H_2(\theta) \geq \epsilon\right] \to 0$$

for all $\epsilon > 0$. Combining the result of Lemma 3.2, we could obtain

$$\mathbb{P}\left[\left|\frac{1}{n}K(X_1X_2\cdots X_n) - H_2(\theta)\right| \geq \epsilon\right] \to 0$$

for all $\epsilon > 0$, i.e. $\frac{1}{n}K(X_1X_2\cdots X_n) \to H_2(\theta)$ in probability. □

Remark. The self-delimiting version of Lemma 3.3 is $\frac{1}{n}K'(X_1X_2\cdots X_n|n) \to H_2(\theta)$ in probability.[18]

Hammer, Romashchenko, Shen, and Vereshchagin (2000) showed that exactly the same linear inequalities hold for both Shannon's entropy and Kolmogorov complexity (up to a logarithmic error term). This explains why all results in Section 2 have their Shannon's entropy counterparts.

Theorem 3.4. (Romashchenko) Any linear inequality for Shannon's entropy is true if and only if it is also true for Kolmogorov complexity (up to a logarithmic term).

It is worth noting that Romashchenko utilized the result we have presented in Lemma 3.3 to demonstrate the part that any form of linear inequality of Kolmogorov complexity holds for Shannon's entropy. [19]

In that sense, Lemma 3.3 reveals an essential connection between the two measures.

4 Conclusion

In this report, we have introduced the concept and properties of Kolmogorov complexity (and its self-delimiting version). Furthermore, we delve into the connections between Kolmogorov complexity and Shannon's entropy, which are two conceptually different measures that, interestingly, appear to have similar characteristics and closely related numerical values. Overall, we have explored one specific part of Kolmogorov complexity's information-theoretic aspect.

However, Kolmogorov complexity is a rich theory, encompassing a broad range of mind-fascinating topics beyond those covered in this report. For instance, the uncomputability of Kolmogorov complexity[20], the connection between (algorithmic) randomness and incompressibility [21], and the notion of universal probability that assigns an a priori probability to objects, which serves as a basis of inductive inference... Further exploration of the concepts and properties of Kolmogorov complexity and the algorithmic information theory would promise to reveal even more interesting insights into the nature of information and computation.

[18]See Cover's *Elements of Information Theory*, Theorem 14.5.3, of which our proof for Lemma 3.3 is modified from.

[19]Readers who are interested may refer to his paper: *Inequalities for Shannon Entropy and Kolmogorov Complexity.*

[20]The uncomputability of K can be shown via a contradiction similar to Berry's paradox: suppose K is computable, then we can devise a program p_C that generates, in shortlex order, all binary strings $s \in \{0,1\}^*$ until $K(s) > C$ where C is a constant s.t. $l(p_C) \leq C$ (since C can be encoded in $O(\log C)$ bits, we could always find a sufficiently large C). However, since $\mathcal{U}(p_C) = s$, we have $K(s) \leq l(p_C) \leq C$, a contradiction.

[21]For example, the sequence "01010101 . . . 01" seems to be less "random" than some other sequences of the same length even if they occur in the same probability as outcomes of tossing a fair coin. We can use Kolmogorov complexity to characterize the "incompressibility" of binary sequences (i.e. x is incompressible by c if $K(x) \geq l(x) - c$), and say that an infinite sequence is Martin-Löf random iff. there exists a constant c such that all prefixes of the sequence are incompressible by c.

References

- Paper, *A Theory of Program Size Formally Identical to Information Theory*, Gregory Chaitin
- Paper, *Inequalities for Shannon Entropy and Kolmogorov Complexity*, Daniel Hammer, Andrei Romashchenko, Alexander Shen and Nikolai Vereshchagin
- Book chapter 14, *Elements of Information Theory 2nd Edition*, Thomas Cover
- Book chapter 4, *Handbook of Theoretical Computer Science (Vol. A): Algorithms and Complexity*, *Kolmogorov Complexity and Its Applications*, Ming Li and Paul Vitányi (edited by J. van Leeuwen)
- Lecture note, *Algorithmic Information Theory and Kolmogorov Complexity*, Alexander Shen
- Lecture note from *COS597D: Information Theory in Computer Science*, Mark Braverman, Princeton University