Kolmogorov complexity

This handout is based on Handout 5 from Luca Trevisan's Spring 2015 CS172 classes, available at `http://www.eecs.berkeley.edu/~luca/cs172/notek.pdf`. It also draws from Section 6.4 of Sipser; the material on palindromes is based on exercise 2.8.5 of Papadimitriou.

We intuitively feel that a long string of zeroes contains less information than a string of the same length containing some complicate pattern of zeroes and ones. One way to try to formalize this intuition is to look at the shortest possible description of the string. A long string of zeroes can be described succinctly as, "One million zeroes", whereas there might not be a better way of describing a more complicated string than just writing out all the characters.

If we're going to use "descriptions" of strings, we need to decide what constitutes a description. Kolmogorov complexity describes strings as the outputs of computations.

1 Definitions and basic properties

Definition 51. A *representation* of a binary string x is any string $\langle M\rangle w$ such that, when Turing machine M runs with input w, it halts with its tape containing exactly the string x. The *Kolmogorov complexity* of x is

$$K(x) = \min\{|u| \mid u \text{ is a representation of } x\}\,.$$

You can think of this in terms of data compression. We have a file x which we've compressed to give a file w. To decompress this, we use the program M and we're interested in the total size of M and w being as small as possible.

We might expect that the shortest description of any string can't be longer than the string itself. In fact, descriptions can be slightly longer than the string, but only by an additive constant.

Lemma 52. *There is a constant c such that, for all x, $K(x) \leq |x| + c$.*

Proof. Let M be any Turing machine that halts immediately without changing its input. Then, for any string x, $\langle M\rangle x$ is a representation of x, so $K(x) \leq |x| + |\langle M\rangle|$. □

One might hope to use a scheme that said, "If there is a representation of x that is shorter than x, use that as a description of x; otherwise, use x to describe itself." In other words, you won't bother compressing your file if doing so wouldn't actually save any space. However, this does not work because, given a string $\langle M\rangle w$, we would not know whether it was supposed to describe literally the string $\langle M\rangle w$ or was supposed to describe the string that the machine M computes when run with input w. We would need to add at least one character to the string to tell the difference between these two cases (e.g., say that $0\langle M\rangle w$ describes the literal string $\langle M\rangle w$, whereas $1\langle M\rangle w$ describes the string produced by M from input w).

At this point, you may be worried that the definition of Kolmogorov complexity appears to depend on the details of the encoding scheme we chose for Turing machines. It does, but not in any significant way.

Lemma 53. *Let $[\cdot]$ be any encoding of Turing machines and let K' denote Kolmogorov complexity with respect to this encoding. As long as the function that maps $[M]$ to $\langle M\rangle$ is computable,*[1] *there is a constant c such that, for all x, $K(x) \leq K'(x) + c$.*

Proof. Let $[M]w$ be a shortest representation of x with respect to the encoding $[\cdot]$.

There is a universal Turing machine T for the encoding $[\cdot]$: use the computable mapping from $[M]$ to $\langle M\rangle$ and then simulate the universal Turing machine for $\langle\cdot\rangle$ (Theorem 39). But now, $\langle T\rangle[M]w$ is a description of x with respect to the encoding $\langle\cdot\rangle$. This description says "Run the universal machine T with input $[M]w$," which is equivalent to "run M with input w", as required.

So, we have

$$K(x) \leq |\langle T\rangle[M]w| = |\langle T\rangle| + K'(x)\,,$$

which proves the lemma, since $|\langle T\rangle|$ is a fixed constant, independent of x. □

The following two bounds don't depend on the encoding in any way. The first one also effectively says that no data compression scheme can compress all data. The second is stronger, saying that any data compression scheme can compress only a tiny fraction of files by even c bits.

Lemma 54. *For every $n \geq 0$, there is a string $x \in \{0,1\}^n$ such that $K(x) \geq n$.*

Proof. There are 2^n binary strings of length n. However, the number of strings of length less than n is $\sum_{i=0}^{n-1} 2^i = 2^n - 1$. So, even if every string of length less than n was a representation of a string of length n, there would not be enough representations to allow $K(x) < n$ for all $x \in \{0,1\}^n$. □

Lemma 55. *For every n and c, the proportion of n-bit strings x such that $K(x) < n - c$ is less than 2^{-c}.*

Proof. Let $S = \{x \in \{0,1\}^n \mid K(x) < n - c\}$. Then

$$\begin{aligned}|S| &\leq \big|\{\langle M\rangle w \mid |\langle M\rangle w| < n - c\}\big| \\ &\leq \big|\{y \in \{0,1\}^* \mid |y| < n - c\}\big| \\ &= 2^{n-c} - 1\,.\end{aligned}$$

(Note that the second inequality does not depend on the encoding scheme used.)

So the proportion of n-bit strings in S is

$$\frac{|S|}{2^n} < \frac{2^{n-c}}{2^n} = 2^{-c}\,.$$ □

So, for example, if we pick a random 1024-bit string by tossing a fair coin for each bit, the probability that its Kolmogorov complexity is less than 1000 is less than 1/10 000 000. Data compression works in practice because the data we are trying to compress is far from random. English text, for example, is highly structured: a random sequence of letters is unlikely to be a word.

Say that a string x is *incompressible* if $K(x) \geq |x|$.

Theorem 56. *The set U of incompressible strings is undecidable*

[1] And, if it is not, you can't even tell what Turing machine is denoted by $[M]$, so the encoding is useless.

Proof. By Lemma 54, there is at least one incompressible string of every length. For each $n \geq 0$, let s_n be the lexicographically first length-n string in U.

Suppose for contradiction that U is decided by some Turing machine M. Then there is a Turing machine M' which, given input $\mathrm{bin}(n)$, the binary representation of n, outputs s_n: the machine tries every string of length n in lexicographic order and simulates M to see if each is in U. Therefore, $K(s_n) \leq |\langle M' \rangle \, \mathrm{bin}(n)| \leq |\langle M' \rangle| + \log n + 1$.

But, the fact that $s_n \in U$, $n \leq K(s_n)$. Therefore, we have $n \leq |\langle M' \rangle| + \log n + 1$, which is a contradiction for large enough n. □

Unsurprisingly, since Kolmogorov complexity depends on the behaviour of Turing machines, most properties related to it are undecidable. Nonetheless, we shall see in the following sections that Kolmogorov complexity and, especially the concept of incompressible strings, can have applications.

2 Application: unprovable statements

While studying Gödel's incompleteness theorems, we came across mathematical statements that are true but cannot be proven within any given proof system. More specifically, we showed that there are statements S such that neither S nor $\neg S$ has a proof. From the proof of Theorem 49, we can read off such a statement: it is the statement that a particular Turing machine terminates when started with its own description as its input.

In this section, we show how to generate at random statements that are true with very high probability but which have no proof.

Theorem 57. *Consider any consistent proof system that includes the axioms of first-order arithmetic and can express the property that $K(x) \geq t$ in a statement which is computable, given x and t. For sufficiently large t, every statement of the form $K(x) \geq t$ is unprovable.*

Proof. Consider the following Turing machine M, taking as its input $\mathrm{bin}(t)$.

- For each $m \geq 1$ in turn,
  - For all strings x and P with length $\leq m$, if P proves that $K(x) \geq t$, then output x and halt.

If $K(x) \geq t$ is provable for some string x, then M will find such a string and output it. But then we have

$$t \leq K(x) \leq |\langle M \rangle \, \mathrm{bin}(t)| \leq |\langle M \rangle| + \log t + 1 \,,$$

and this is a contradiction for sufficiently large t. □

This theorem gives us a way to generate statements that are, with high probability, both true and unprovable. If we fix all the details of a formalization of mathematics, then the Turing machine M in the proof is completely defined. Then, we just need to find some t such that $t > |\langle M \rangle| + \log t + 1$ and no statement of the form $K(x) \geq t$ is provable. For example, if we pick a string x of length at least $t + c$ uniformly at random from all such strings, then the statement $K(x) \geq t$ is unprovable by Theorem 57, but is true with probability at least $1 - 2^{-c}$ by Lemma 55.

3 Application: lower bounds

One way for a single-tape Turing machine to decide the language of palindromes is as follows. Go to the first character of the input, remember it and replace it with a blank. Then, find the last character: if it matches, replace it with a blank and repeat; if it doesn't match, reject. For an input of length n, this takes $n/2$ scans to the right and $n/2$ scans back to the left. The scans have average length $n/2$ so the total time taken is about $n^2/2$ steps. This feels rather inefficient, given that we can solve the problem in linear time using random-access memory. However, in this section, we show that any single-tape Turing machine that decides the language of palindromes must require $\Omega(n^2)$ steps, so the naïve algorithm is optimal up to constant factors.

Consider a Turing machine M running on input w. Say that, at step t, M *crosses tape cell* i if, at that step, either it moves from cell i to $i+1$ or from $i+1$ to i. Define the *crossing sequence for cell* i to be the (finite or infinite) sequence of pairs $(q_1, a_1), (q_2, a_2), \ldots$ such that, the transition of M that crosses i for the jth time is caused by being in state q_j and reading symbol a_j. Note that, if j is odd, then M read a_j at cell i and moved right; if j is even, it read a_j at cell $i+1$ and moved left.

We can view the crossing sequence for cell i in the following terms. Suppose we observe M operating on input w but we only look at the first i tape cells. M starts to work within this window and eventually leaves it because it read a_1 while in state q_1. It then operates outside the window. We can't see what it does but, whatever happens only depends on what was written on the tape outside the window and the fact that the machine left the window in state $\delta(q_1, a_1)$. After a while the head returns to the window with the machine in state $\delta(q_2, a_2)$. It then stays inside the window for a while, before leaving in state $\delta(q_3, a3)$ and so on.

Theorem 58. *Every Turing machine that decides the language of palindromes takes $\Omega(n^2)$ steps for inputs of length n.*

Proof. Suppose M decides the language of palindromes and consider its computation on the string $x0^n x^{\mathrm{R}}$ for any string x of length n. (x^{R} denotes the reversal of x.) Suppose this computation terminates after $T(x)$ steps.

There must be some $i \in \{n+1, \ldots, 2n\}$ such that the crossing sequence for cell i has length $m \leq T(x)/n$: if not, then the computation must visit each of the n zeroes more than $T(x)/n$ times, so the computation takes more than $T(x)$ steps. Let this crossing sequence be $(q_1, a_1), \ldots, (q_1, a_m)$.

Now, x is the unique string such that $x0^{i-n}$ has the following property. If M receives input $x0^{i-n}$, the first crossing of cell i is (q_1, a_1). If the head were then placed immediately back at cell i with the machine in state q_2, the next crossing of i would be (q_3, a_3), and so on. x must be unique because, if $y \neq x$ had the same property, then M would accept $y0^n x^{\mathrm{R}}$, which is not a palindrome.

Therefore, the string x can be described by giving the description of a Turing machine M', the number $n = |x|$, the number i and the crossing sequence $(q_1, a_1), \ldots, (q_m, a_m)$. The machine M' operates as follows. In turn, for each string y of length n, it constructs the string $y0^{i-n}$. It then simulates M and checks whether its execution has the property described above. If so, it outputs y, and $y = x$.

We have shown that, for an arbitrary binary string x, $K(x) \leq |\langle M' \rangle| + cT(x)/n + 2\log n$, where c is the number of bits required to write the details of a crossing. Now, suppose that $T(x) = o(|x|^2)$. Then, for all sufficiently long x, $K(x) < |x|$, contradicting Lemma 54. □