Lecture 9:
Kolmogorov complexity

Valentine Kabanets

October 20, 2016

1 Kolmogorov Complexity

Usually, when we say that a string x is random, we have some probability distribution in mind and we mean that x is chosen according to this probability distribution. Then we can talk about the amount of randomness (information) contained in a given probability distribution. This can be measured by Shannon's notion of *entropy*, which corresponds to a minimal average description length of the given distribution.

In contrast, Kolmogorov's notion of descriptive complexity refers to a single given string (no distribution is involved). In an application of computability theory to randomness, Kolmogorov defined the notion of *descriptive complexity* of a finite binary string x as the length of a *shortest* pair $d(x) = \langle M, w\rangle$, where TM M on input w outputs x. Here the pair $\langle M, w\rangle$ is encoded as $\langle M\rangle w$, where $\langle M\rangle$ is a "repetition" code (each bit is repeated twice), followed by 01 to mark the end of $\langle M\rangle$. We denote the length $|d(x)|$ by $K(x)$.

A given string x is considered random, *Kolmogorov-random*, if its shortest description $d(x)$ is not shorter than its length $|x|$. Such strings are also called incompressible. This corresponds to our intuition that a string looks random to us if we cannot compress it (cannot see some pattern that would allow us to compress the string).

Thus, according to Kolmogorov,

Random = Incompressible.

Theorem 1. *For every n, there is a string x of length n such that x is incompressible.*

Proof. (by counting): The number of strings of length n is 2^n. On the other hand, the number of descriptions of length $< n$ is $2^0 + 2^1 + 2^2 + ... + 2^{n-1} = 2^n - 1$, which is less than 2^n. Hence there is at least one incompressible string of length n. □

1.1 Some properties of Kolmogorov complexity

First, every string has Kolmogorov complexity at most the length of that string, plus a constant.

Theorem 2. *There is a constant c such that for all strings x, $K(x) \leq |x| + c$.*

Proof. Let M be a TM that halts as soon as it is started. Then $\langle M\rangle x$ is a description of x, and so $K(x) \leq |\langle M\rangle| + |x| = |x| + c$. □

Secondly, if a string is repeated twice, the Kolmogorov complexity of the new string stays the same, modulo an additive constant. Also, the complexity of xy is approximately the sum of the complexities of x and y separately.

Theorem 3. *1. there is a c such that for all x, $K(xx) \leq K(x) + c$.*

2. there is a c such that for all x, y, $K(xy) \leq 2K(x) + K(y) + c$.

Finally, while the notion of descriptive complexity was defined with respect to Turing machines, it is in fact *independent* of the model of computation we may choose. Fix any computable function p from strings to strings. Define $d_p(x) =$ lexicographically shortest string s such that $p(s) = x$. Let $K_p(x) = |d_p(x)|$.

Theorem 4. *For all computable functions p, there is a constant c such that, for every x, $K(x) \leq K_p(x) + c$.*

Proof. Define M: "On input w, output $p(w)$." Then $\langle M \rangle d_p(x)$ is a description of x, and its length is $c + K_p(x)$. □

1.2 Hard problems related to Kolmogorov complexity

Computing $K(x)$ on every given $x \in \{0,1\}^*$ is an impossible task. Similarly, it's impossible to have an algorithm for deciding (or even semi-deciding) if a given string x is Kolmogorov-random.

Theorem 5. *1. The Kolmogorov-complexity function $K : \{0,1\}^* \to \{0,1\}^*$ is not computable.*

2. The language $\{x \in \{0,1\}^* \mid K(x) \geq |x|\}$ is not semi-decidable.

The proof of this theorem is left as a (homework) exercise!

1.3 Kolmogorov randomness

The claimed equivalence between Random strings and Incompressible strings may be justified in many cases. For example, it can be shown that incompressible strings have many properties of truly random strings (such as about the same number of zeros and ones, a string of length n has $O(\log n)$ of consecutive zeros, etc.) Next we'll see that incompressible strings will satisfy any computable property that holds for almost all strings (so in some very precise sense, an incompressible string behaves like a "typical" string).

Theorem 6. *Given any computable function $f : \{0,1\}^* \to \{0,1\}$ such that*

$$|\{x \mid |x| = n,\ f(x) = 0\}| < 2^{o(n)},$$

every incompressible string y (of sufficiently large length n) satisfies f, i.e., $f(y) = 1$.

Proof. The idea of the proof is the following. Each string x in the set of all n-bit strings rejected by the function f (call this set Bad) can be uniquely described by its index in that set. Since the set Bad is small, and since each string in Bad has index of bit-length at most $\log |Bad| \leq o(n)$, we get that each n-bit string $x \in Bad$ has a description of length strictly less than n (namely, at most $o(n) + c$, where c is the size of the Turing machine computing the function f, as we need to know f in order to specify a string number i rejected by f). Thus, every bad string x is *compressible*. It follows that every incompressible string y will satisfy f. □

Thus, in some very precise sense, if a computable property is true on most binary strings, it will also be true on Kolmogorov-incompressible strings.

If we could generate Kolmogorov-incompressible strings efficiently deterministically, then we could convert every efficient randomized algorithm into an equivalent efficient deterministic algorithm. Unfortunately, as you're asked to show in your homework, there is no algorithm for generating Kolmogorov-incompressible strings (let alone an efficient algorithm). Nonetheless, using a scaled-down version of Kolmogorov-complexity, namely Boolean circuit complexity, of a binary string, it is possible to show that binary strings of high circuit complexity can be used to efficiently *derandomize* randomized algorithms. Thus, if we can efficiently generate binary strings of high circuit complexity, then we can efficiently derandomize all randomized algorithms.

While generating strings of high circuit complexity is possible (unlike generating high Kolmogorov-complexity strings), it is still unknown how to generate such strings efficiently. In fact, this problem is related to the very difficult open questions in complexity theory of proving circuit lower bounds for explicit functions, which in turn is related to the famous "P versus NP" problem, which we study later in the course.