Kolmogorov Complexity

Introduction

- Quantifies the intrinsic descriptive complexity of an object.
- If X is a random variable, there is a sense to say that the descriptive complexity of the event $X = x$ is $\log \frac{1}{p(x)}$, as that is the codeword length by Shannon code.
- Kolmogorov defines the algorithmic (descriptive) complexity of an object (represented in bits, say) to be the length of the shortest binary computer program that prints out the object.

Example of Descriptive Complexity

- Consider the three bit strings as given in pp 145.
  - The first string is a repetition of 01.
  - The second looks random but it is the binary expansion of $\sqrt{2}-1$
  - The third string is random but the proportion of 1's is not 0.5
- The first and second strings are simple to describe, while the third is not. By describing the number of 1's and giving the index of this string in the set of strings with the same number of 1's, one needs $\log n + nH(\frac{k}{n})$ bits.

Models of Computation

- Universal Turing machine
  - Finite state control
  - Input tape: stores program (and initial data)
  - Work tape: stores transitory data
  - Output tape
- Here we assume that the machine reads program from right to left and not backwards. Then the set of programs that halt is a prefix-free set.

Definition of Kolmogorov complexity

- The Kolmogorov complexity $K_U(x)$ of a string x w.r.t. a universal computer is defined as

  $$K_U(x) = \min_{p:U(p)=x} l(p),$$

  where $l(p)$ is the length of p, which prints x and halts.

Universality of Kolmogorov Complexity

- Let U be a universal computer and A be any other computer, then

$$K_U(x) \leq K_A(x) + c_A,$$

  where c_A does not depend on x.

- c_A is basically the length of a program on U which simulates A with U.

- It is Ok to focus on universal machine, so we can drop the U subscript.

Conditional Kolmogorov Complexity

- The conditional Kolmogorov complexity $K(x)$ knowing the length of a string x is defined as

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

- Theorem

$$K(x|l(x)) \leq l(x) + c,$$

since we can use the program:
Print out the following l-bit sequence $x_1 x_2 \ldots$

Upper Bound on Kolmogorov Complexity

- If the length of x is not known to the machine, we can describe $l(x)$ by repeating twice every digit of the binary representation of $l(x)$, and then appending 01. The extra number of bits is $2\log l(x) + 2$, so

$$K(x) \leq K(x|l(x)) + 2\log l(x) + c.$$

- In fact the upper bound can be improved to be

$$K(x) \leq K(x|l(x)) + \log^* l(x) + c.$$

Examples of Kolmogorov Complexity

- The sequence of n zeros

$$K(000\ldots0|n) = c$$

- The sequence of the first n bits of π

$$K(\pi_1\pi_2\ldots\pi_n|n) = c$$

- An integer n:

$$K(n) \leq \log^* n + c$$

Kolmogorov Complexity of A Binary String

- The sequence of n bits with k ones:

  Generate, in lexicographic order, all sequences with k ones. Of these sequences, print the ith sequence.

  $$l(p) = c + 2\log k + \log C_k^n \leq c + 2\log k + nH_0(\frac{k}{n}),$$

  where H_0 is the binary entropic function.

- It follows that

  $$K(x_1 \ldots x_n|n) \leq c + 2\log n + nH_0(\frac{1}{n}\sum x_i)$$

- A string is compressible if $K(x) < l(x)$.

Kolmogorov Complexity and Entropy

- Let an i.i.d. process $\{X_1, X_2, \dots\}$ with a finite alphabet $\mathcal{X}$ be drawn according to $f(x)$. Let $f(x^n) = \prod_i f(x_i)$. Then

$$H(X) \le \frac{1}{n} \sum_{x^n} f(x^n) K(x^n|n) \le H(X) + \frac{|\mathcal{X}| \log n}{n} + \frac{c}{n},$$

  for some c.

- It follows that

$$E \frac{1}{n} K(X^n|n) \to H(X)$$

Proof for Lower Bound

- The halting programs satisfy the prefix property, thus the lengths satisfy the Kraft's inequality. We assign each x^n to the shortest program that prints it and halts. These shortest programs also satisfy Kraft's and thus the expected length is no less than the entropy. Hence

$$\sum_{x^n} f(x^n) K(x^n|n) \geq H(X_1, \ldots, X_n) = nH(X),$$

Proof for Upper Bound

- We have established, for binary sequences,

$$K(x_1 \ldots x_n|n) \leq c + 2\log n + nH_0(\frac{1}{n}\sum x_i).$$

Hence

$$\begin{aligned} EK(X_1 \ldots X_n|n) &\leq c + 2\log n + nEH_0(\frac{1}{n}\sum X_i) \\ &\leq c + 2\log n + nH_0(\frac{1}{n}\sum EX_i) \\ &= c + 2\log n + nH_0(p). \end{aligned}$$

- For $|\mathcal{X}| > 2$, we can use the method of types, which will be described later.

Kolmogorov Complexity of Integers

- Definition

$$K(n) = \min_{p:U(p)=n} l(p).$$

- Upper bound

$$K(n) \leq \log^* n + c.$$

- There is an infinite number of n such that

$$K(n) > \log n.$$

See the text for proof.

Probability of Simple Sequences

- There are long strings and large numbers that are simple to describe.
- However, most sequences or numbers do not have simple (short) descriptions.
- Let $X_1 \ldots X_n$ be drawn from Bernoulli$(\frac{1}{2})$. Then

$$p(K(X_1 \ldots X_n|n) < n - k) < 2^{-k}.$$

  See the text for proof.
- Thus most sequences have a complexity close to their length.

Random and Incompressible Sequences

- A sequence is algorithmically random if

$$K(x_1 \dots x_n | n) \geq n.$$

- An infinite string is incompressible if

$$\lim_{n \to \infty} \frac{K(x_1 \dots x_n | n)}{n} = 1.$$

- If $x_1, x_2, \dots$ is incompressible, then

$$\frac{1}{n} \sum_i x_i \to \frac{1}{2}.$$

That is, the proportions of 0 and 1 are almost equal.

Universal Probability

- The universal probability of a string x is

$$P_U(x) = \sum_{p:U(p)=x} 2^{-l(p)} = Pr(U(p) = x).$$

  which is the probability that a random program (drawn i.i.d.) prints out the sequence x.

- Imagine a monkey sitting in front of a computer and typing keys at random to create a program. Will the output looks random?

- Since shorter programs are more probable, simpler strings are more likely to be printed out than complicated ones.

Halting Problem

- For any computational model, there is no general algorithm to decide whether a program will halt or not.
- So there are problems that are computable by a computer and there are problems that are not.
- One consequence of interest is that the Kolmogorov complexity is non-computable.
  - The only way to find the shortest program is to test the short programs until one is found.
  - But it may not be found.

Ω

- Definition

  $$\Omega = \sum_{p:U(p) \text{ halts}} 2^{-l(p)}$$

  It is the probability that a program randomly drawn i.i.d. from Bernoulli($\frac{1}{2}$) halts.

- Property
  - $\Omega \leq 1$ since halting programs are prefix-free.
  - Ω is not computable (as halting problem is not).
  - Ω is not compressible (Theorem 7.8.1).

Running All Programs in Parallel

- It is possible to run all programs in parallel with the scheme as outlined in the text.
- A program that halts will eventually halt and its contribution to Ω can be recorded.
- Knowing the first n bits of Ω allows us to determine the whether a program of length $\leq n$ bits halts: Simply run all programs in parallel until the sum exceeds Ω_n. All programs with length $\leq n$ not halting yet will not halt.

Incompressibility of Ω

- **Theorem:** There exists a constant c such that

$$K(\omega_1 \ldots \omega_n) \geq n - c, \ \ \forall n.$$

- **Proof:** Using $K(\omega_1 \ldots \omega_n)$ bits, we can print and know the first n bits of Ω, which enables us to know which programs with length $\leq n$ bits halt together with their outputs. Find a string x_0 that is not in the output list, then the complexity of x_0 is greater than n, i.e. $K(x_0) > n$. Furthermore, $K(x_0) \leq K(\Omega_n) + c$ since x_0 is described by the routine to compute Ω_n appended by a routine of finding and printing x_0. Thus the theorem.

Universal Gambling

- A gambler is gambling sequentially on a random binary sequence with 2-for-1 odds.
- Suppose the gambler bets by universal gambling, that is

$$b(x) = 2^{-K(x)}.$$

Note that

$$\sum b(x) = \sum 2^{-K(x)} \leq \Omega \leq 1.$$

Gambler's Wealth

- The complexity, the wealth and the length of x is related by

$$\log S(x) + K(x) \geq l(x).$$

which follows from

$$S(x) = \sum_{x' \text{prefixed by} x} 2^{l(x')} b(x') \geq 2^{l(x)} 2^{-K(x)}$$

- For sequence with finite complexity

$$S(x) \geq 2^{l(x)-c},$$

which is asymptotically equivalent to knowing the sequence in advance (wealth is $2^{l(x)}$).