AIT and Solomonoff Induction: a brief crash course

20.6.2019 ↓ 21.6.2019

Li, Vitányi, An Introduction to Kolmogorov Complexity and its Applications, 3rd ed. (Springer, 2008)

Necessary background: computability, Turing machines etc.
→ will instead appeal to intuition.

Binary strings: $\{0,1\}^* = \{\varepsilon, 0, 1, 00, 01, 10, 11, 000, \ldots\}$

Partial recursive function $\phi : \{0,1\}^* \to \{0,1\}^*$:
think of programming language (C, LISP, ...) running on an unbounded computer, mapping input x to output $\phi(x)$ (sometimes crashes → $\phi(x)$ undefined, *partial* fn.)

Can encode pairs of strings (x,y) into single strings:
e.g. $\underbrace{11\ldots1}_{\ell(x)} x y =: \langle x,y\rangle \in \{0,1\}^*$. → define $\phi(x,y) := \phi(\langle x,y\rangle)$.

Similarly, can "input more than two strings into the computer".

1. Plain and prefix Kolmogorov Complexity

Definition: Given any PRF ϕ, define

$$C_\phi(y|x) := \min\{\ell(p) \mid \phi(p,x) = y\}.$$

"shortest program to compute y from x".

$$C_\phi(y) := C_\phi(y|\varepsilon)$$ "length of shortest program to compute x".

Theorem: There exists a ("universal") PRF U such that

$$C_U(y|x) \leq C_\phi(y|x) + c_\phi \quad (c_\phi \in \mathbb{N} \text{ additive constant}).$$

Intuition: U is a "universal progr. language" (like C, LISP) which, on input

- description of PRF ϕ
- input p to ϕ

simulates ϕ on input p and generates the corresponding output.

Then c_ϕ = length of description of ϕ.

$\Rightarrow$ If U and V are both universal, $\boxed{C_U(y|x) = C_V(y|x) + O(1)}$.

"Invariance of Kolmogorov Complexity" Set $C(y|x) := C_U(y|x)$.

In AIT, one doesn't care about additive constants.

<u>Example:</u> $C(x) \leq l(x) + O(1)$.

<u>Proof:</u> $\phi(x) := x$, PRF that copies input to output $\Rightarrow C_\phi(x) = l(x)$

$\Rightarrow C(x) = C_U(x) \leq C_\phi(x) + c_\phi$. $\square$

Discuss <u>Examples</u>: $C(xx) \leq C(x) + O(1)$

$$C(\underbrace{111\ldots1}_{n}) \leq \log_2 n + O(1)$$
$$= C(n) + O(1)$$

[redacted] $C(\pi_{1:n} | n)$ [redacted] $= O(1)$

$\pi_{1:n}$: first n digits of $\pi = 3.14159\ldots$

Is $C(x,y) \leq C(x) + C(y) + O(1)$?

$\overbrace{\text{run both:}}^{O(1)\text{ bits}}$	program for x	program for y

= program for (x,y) ?

Problem: don't know where program for x ends. So, <u>NO</u>!

$\Rightarrow$ define $K(y|x)$ as shortest length of <u>self-delimiting</u> program that computes y from x

0	1	0	1	1	0	0	1	0	1	...

input — random junk

$\to$ machine needs to know "where input ends".

$\Rightarrow K(x,y) \leq K(x) + K(y) + O(1)$.

$$K(x) \leq \ell(x) + K(\ell(x)) + O(1)$$

Most strings are random, i.e. incompressible:
For each fixed $r \in \mathbb{N}$, the number of $x \in \{0,1\}^n$ with $K(x) \leq n + K(n) - r$ does not exceed $2^{n-r+O(1)}$

2. One version of algorithmic probability

Idea: obtain one bit after the other $(0,1,0,0,\ldots)$ probabilistically

Prob. measure P: $P(\varepsilon) = 1$, $P(x0) + P(x1) = P(x)$.

E.g. repeated coin toss: $P(x) = \left(\frac{1}{2}\right)^{\ell(x)}$.

Semimeasure: $m(\varepsilon) \leq 1$, $m(x0) + m(x1) \leq m(x)$.

Monotone machine T: runs indefinitely.

input | 0 | 1 | 0 | 0 | ... $\longrightarrow$ read bits sequentially

output | 1 | 1 | 1 | ... $\longrightarrow$ write bits sequentially

Write $T(p) = x*$ if, after reading p (and not more!), the machine output starts with x.

$$\text{Set} \quad M_T(x) := \sum_{p : T(p) = x*} 2^{-\ell(p)}$$

This is a semimeasure. Interpretation: It's the prob. that, on coin-tossing random input bits, T will output x (and perhaps more).

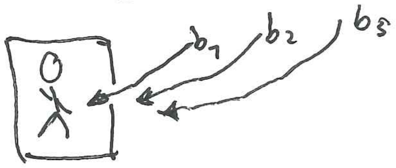
$\boxed{\text{Theorem: There exists a "universal" monotone machine } U \text{ such that } M_U(x) \geq M_T(x) \cdot c_T \text{ for all MM. } T, \text{ where } c_T \in \mathbb{R}^+.}$

Set $M(x) := M_U(x)$. Conditional prob. of next bit, given previous ones:

$$M(0|x) := \frac{M(x0)}{M(x)}, \quad M(1|x) := \frac{M(x1)}{M(x)}.$$

Outlook/Exercise: The M_u are exactly the universal enumerable semimeasures. Obtain same class if input not chosen via iid coin toss, but ~ other computable method.

Setting for Solomonoff induction: Agents receives bits $b_1, b_2, b_3, \ldots$



They are distributed according to some unknown computable measure μ.

Task: agent has to estimate the actual prob. $\mu(b|x_1^n)$ of the next bit $b = b_{n+1}$ given [redacted] $x_1^n := b_1 b_2 \ldots b_n$.

Computability of μ: There exists a computer program which, on inputs $x \in \{0,1\}^*$ and $m \in \mathbb{N}$, outputs a $(1/m)$-approximation to $\mu(x)$.

Theorem: Let μ be a computable measure. Then there is a set $S \subseteq \{0,1\}^\infty$ of μ-measure 1 such that for every $x \in S$

"universal induction"

$$|M(0|x_1^n) - \mu(0|x_1^n)| + |M(1|x_1^n) - \mu(1|x_1^n)| \xrightarrow{n\to\infty} 0.$$

Discuss! How can the agent achieve its task?

(Practical problem: M is not computable.)

- if x_1^n is very long and random (~ coin tossing), then $M(b|x_1^n) \approx \frac{1}{2}$.
- Suppose all your previous bits are $b_1 = b_2 = \ldots = b_n$ (perhaps because $\mu(1^m) = 1$ for every m, i.e. deterministic process)

$$M(0|1^n) = 2^{-K(n) + O(1)} \xrightarrow{n\to\infty} 0.$$

For "most" n, this is $\approx 2^{-\log(n) + O(1)} \approx 1/n \cdot O(1)$.

but e.g. $n = 10^{10^{10^{\cdot^{\cdot^{\cdot}}}}}$ (power tower of height 10^{100}) has $K(n)$ very small

discuss!