Kolmogorov Complexity 101

Note: actually this slides has only 7 pages

How much information does a random coin flip have?

How much information does a random coin flip have?

$H = \frac{1}{2}\log 2 + \frac{1}{2}\log 2 = 1 \text{ bit}.$

How much information does a random coin flip have?

$$H = \frac{1}{2}\log 2 + \frac{1}{2}\log 2 = 1 \text{ bit.}$$

How much information does *War and Peace* have?

How much information does a random coin flip have?

$H = \frac{1}{2}\log 2 + \frac{1}{2}\log 2 = 1 \text{ bit.}$

How much information does *War and Peace* have?

$\text{Novel}\colon \Omega \to [0,1], \Omega = 2^{L^*}, L = \{a, b, \dots\}$?

How much information does a random coin flip have?

$H = \frac{1}{2}\log 2 + \frac{1}{2}\log 2 = 1 \text{ bit}.$

How much information does *War and Peace* have?

$\text{Novel}\colon \Omega \to [0,1], \Omega = 2^{L^*}, L = \{a, b, \dots\}$?

possibly by a $\text{Write}\colon (L^n, L) \to [0,1]$?

How much information does a random coin flip have?

$H = \frac{1}{2}\log 2 + \frac{1}{2}\log 2 = 1 \text{ bit.}$

How much information does *War and Peace* have?

$\text{Novel}\colon \Omega \to [0,1], \Omega = 2^{L^*}, L = \{a, b, \dots\}$?

possibly by a $\text{Write}\colon (L^n, L) \to [0,1]$?

More reasonable: the complexity of (information in) $x \in S$ is its *shortest description*.

Let's be formal!

Let's be formal!

shortest description of $x \in S$

Let's be formal!

shortest description of $x \in S$

$\to$ shortest description of $n(x)$ where $n\colon S \to \mathbb{N}$ is a unique coding

Let's be formal!

shortest description of $x \in S$

$\longrightarrow$ shortest description of $n(x)$ where $n\colon S \to \mathbb{N}$ is a unique coding

$\longrightarrow$ shortest program to produce $n(x)$ where ...

Let's be formal!

shortest description of $x \in S$

$\longrightarrow$ shortest description of $n(x)$ where $n\colon S \longrightarrow \mathbb{N}$ is a unique coding

$\longrightarrow$ shortest program to produce $n(x)$ where ...

wait!

Let's be formal!

shortest description of $x \in S$

$\rightarrow$ shortest description of $n(x)$ where $n \colon S \rightarrow \mathbb{N}$ is a unique coding

$\rightarrow$ shortest program to produce $n(x)$ where …

wait!

shortest program in which PL?

No one wants to write numeric computing in … any language other than FORTRAN.

— Sussman, father of Scheme, in a talk whose title I can't recall.

Let's be formal!

shortest description of $x \in S$

$\rightarrow$ shortest description of $n(x)$ where $n \colon S \rightarrow \mathbb{N}$ is a unique coding

$\rightarrow$ shortest program to produce $n(x)$ where ...

wait!

shortest program in which PL?

No one wants to write numeric computing in ... any language other than FORTRAN.

— Sussman, father of Scheme, in a talk whose title I can't recall.

or in fancy jargons: shortest input in which Turing machine? can we make a best choice?

Let's be formal!

shortest description of $x \in S$

$\rightarrow$ shortest description of $n(x)$ where $n\colon S \rightarrow \mathbb{N}$ is a unique coding

$\rightarrow$ shortest program to produce $n(x)$ where ...

wait!

shortest program in which PL?

No one wants to write numeric computing in ... any language other than FORTRAN.

— Sussman, father of Scheme, in a talk whose title I can't recall.

or in fancy jargons: shortest input in which Turing machine? can we make a best choice?

Yes.

Theorem 1. *There is a Turing machine* T_0 *which, for any other Turing machine* T*, there exists a constant* c *such that, for any* $n \in \mathbb{N}$*, the shortest program to output* n *(encoded as binary strings) for* T_0 *is at most* c *longer than that of* T*.*

Theorem 2. *There is a Turing machine T_0 which, for any other Turing machine T, there exists a constant c such that, for any $n \in \mathbb{N}$, the shortest program to output n (encoded as binary strings) for T_0 is at most c longer than that of T.*

In short: there is a Turing machine that's as good as any other (within constant difference).

Idea: interpreter, or universal machine if you like it.

Theorem 3. *There is a Turing machine T_0 which, for any other Turing machine T, there exists a constant c such that, for any $n \in \mathbb{N}$, the shortest program to output n (encoded as binary strings) for T_0 is at most c longer than that of T.*

In short: there is a Turing machine that's as good as any other (within constant difference).

Idea: interpreter, or universal machine if you like it.

So we can just define $C(x)$ being the shortest program according to T_0.

Theorem 4. *There is a Turing machine T_0 which, for any other Turing machine T, there exists a constant c such that, for any $n \in \mathbb{N}$, the shortest program to output n (encoded as binary strings) for T_0 is at most c longer than that of T.*

In short: there is a Turing machine that's as good as any other (within constant difference).

Idea: interpreter, or universal machine if you like it.

So we can just define $C(x)$ being the shortest program according to T_0.

Note: for different universal machines the definition only varies by a constant, *if the ordering of Turing machines is fixed*. But it's also true for two effective enumerations.

Theorem 5. *There is a Turing machine T_0 which, for any other Turing machine T, there exists a constant c such that, for any $n \in \mathbb{N}$, the shortest program to output n (encoded as binary strings) for T_0 is at most c longer than that of T.*

In short: there is a Turing machine that's as good as any other (within constant difference).

Idea: interpreter, or universal machine if you like it.

So we can just define $C(x)$ being the shortest program according to T_0.

Note: for different universal machines the definition only varies by a constant, *if the ordering of Turing machines is fixed*. But it's also true for two effective enumerations.

We can similarly define $C(x, y), C(x|y)$ (the latter is interesting).

Some boring basic facts:

Some boring basic facts:

$C(x) \leqslant l(x) + O(1), x \in \mathbb{N}$ where $l(x)$ is the length of the binary representation of x.

Some boring basic facts:

$C(x) \leqslant l(x) + O(1), x \in \mathbb{N}$ where $l(x)$ is the length of the binary representation of x.

$C(x|y) \leqslant C(x) + O(1)$.

Some boring basic facts:

$C(x) \leqslant l(x) + O(1), x \in \mathbb{N}$ where $l(x)$ is the length of the binary representation of x.

$C(x|y) \leqslant C(x) + O(1)$.

Some interesting miserable facts:

Some boring basic facts:

$C(x) \leqslant l(x) + O(1), x \in \mathbb{N}$ where $l(x)$ is the length of the binary representation of x.

$C(x|y) \leqslant C(x) + O(1)$.

Some interesting miserable facts:

$C(x, y) \not\leqslant C(x) + C(y) + O(1)$.

Some boring basic facts:

$C(x) \leqslant l(x) + O(1), x \in \mathbb{N}$ where $l(x)$ is the length of the binary representation of x.

$C(x|y) \leqslant C(x) + O(1)$.

Some interesting miserable facts:

$C(x, y) \nleqslant C(x) + C(y) + O(1)$.

$\exists s, t, C(s \mathbin{+\!\!\!+} t) \ll C(s)$.

Some boring basic facts:

$C(x) \leqslant l(x) + O(1), x \in \mathbb{N}$ where $l(x)$ is the length of the binary representation of x.

$C(x|y) \leqslant C(x) + O(1)$.

Some interesting miserable facts:

$C(x, y) \nleqslant C(x) + C(y) + O(1)$.

$\exists s, t, C(s + t) \ll C(s)$.

They can be derived from

Theorem 12. *in a set S of size n, there are at least $m\,(1 - 2^{-c}) + 1$ elements satisfying $C(x) \geqslant \log m - c$.*

Proof. count the number of programs shorter than $\log m - c$. $\square$

Yes, you can work this out at ten, but sometimes important facts are simple.

Back to information theory

Back to information theory

$\sum_x p(x)\, C(x) \leqslant \sum_x -p(x) \log p(x) + c_p$.

In fact, this is derived from $\sum_x p(x)\, K(x) \leqslant \sum_x -p(x) \log p(x) + c_p$. K is a little different from C and prefix-free.

Back to information theory

$\sum_x p(x)\, C(x) \leqslant \sum_x -p(x) \log p(x) + c_p$.

In fact, this is derived from $\sum_x p(x)\, K(x) \leqslant \sum_x -p(x) \log p(x) + c_p$. K is a little different from C and prefix-free.

We can define $I(x;y) = C(y) - C(y|x)$ like we did before, but …

Back to information theory

$\sum_x p(x)\, C(x) \leqslant \sum_x -p(x) \log p(x) + c_p$.

In fact, this is derived from $\sum_x p(x)\, K(x) \leqslant \sum_x -p(x) \log p(x) + c_p$. K is a little different from C and prefix-free.

We can define $I(x; y) = C(y) - C(y|x)$ like we did before, but ...

Theorem 19. *$C(x, y) = C(x) + C(y|x) + O(\log C(x, y))$. i.e., $C(x) + C(y|x) - c_1 \log C(x, y) \leqslant C(x, y) \leqslant C(x) + C(y|x) + c_2 \log C(x, y)$.*

Corollary 20. *$|I(x; y) - I(y; x)| = O(\log C(x, y))$.*

In other words, I is not symmetric.

Thank you!