Introduction to Theoretical Computer Science

Lecture 17: Information and Descriptive Complexity

Dr. Liam O'Connor

LFCS, University of Edinburgh
CECS, Australian National University
Semester 1, 2023/2024

Informative Examples

Consider the following two binary strings:

Example

$$A = 01010101010101010101010101010101010101$$
$$B = 11100111100000111011100001111001111001$$

Which of A and B contains more *information*?

Informative Examples

Consider the following two binary strings:

Example

$$A = 01010101010101010101010101010101010101$$
$$B = 11100111100000111011100001111001111001$$

Which of A and B contains more *information*?

Applying Compression

The simplest compression algorithm in the world is *run-length encoding*. Applying that gives us:

$$A = 0^1 1^1 0^1 1^1 0^1 1^1 0^1 1^1 0^1 1^1 0^1 1^1 0^1 1^1 0^1 1^1 \ldots$$
$$B = 1^3 0^2 1^4 0^5 1^3 0^1 1^3 0^4 1^4 0^2 1^4 0^2 1^1$$

Now B is shorter!

Informative Examples

Consider the following two binary strings:

Example

$$A = 01010101010101010101010101010101010101$$
$$B = 11100111100000111011100001111001111001$$

Which of A and B contains more *information*?

Applying Compression

The simplest compression algorithm in the world is *run-length encoding*. Applying that gives us:

$$A = 0^1 1^1 0^1 1^1 0^1 1^1 0^1 1^1 0^1 1^1 0^1 1^1 0^1 1^1 0^1 1^1 \ldots$$
$$B = 1^3 0^2 1^4 0^5 1^3 0^1 1^3 0^4 1^4 0^2 1^4 0^2 1^1$$

Now B is shorter! But a smarter compression algorithm could represent A as $[01]^{19}$.

Minimal Length Descriptions

Definition

A *description* of a binary string s is itself a binary string $\langle M, w \rangle$ encoding a pair of Turing machine M and input w, such that when M is executed on w it will output s.

Minimal Length Descriptions

Definition

A *description* of a binary string s is itself a binary string $\langle M, w \rangle$ encoding a pair of Turing machine M and input w, such that when M is executed on w it will output s.

One subtlety is that we cannot just use any pairing function $\langle \cdot, \cdot \rangle$ here, but instead must use one that produces the shortest possible strings.

Pairing

We define $\langle M, w \rangle$ to be the string $\ulcorner M \urcorner w$, that is, the binary encoding of the machine M appended to the string w. The encoding $\ulcorner M \urcorner$ is basically standard, but with some kind of delimiter (see later)

Kolmogorov Complexity

Definition

The *descriptive complexity* $K(s)$ of a string s is the length of the *minimal description* of s,

Kolmogorov Complexity

Definition

The *descriptive complexity* $K(s)$ of a string s is the length of the *minimal description* of s, i.e., the length of the shortest string encoding $\langle M, w \rangle$ such that the machine M run on input w produces s.

Kolmogorov Complexity

Definition

The *descriptive complexity* $K(s)$ of a string s is the length of the *minimal description* of s, i.e., the length of the shortest string encoding $\langle M, w \rangle$ such that the machine M run on input w produces s.

This definition is relatively robust with respect to the type and encoding of our machine M: it differs only by a constant factor.

Kolmogorov Complexity

Definition

The *descriptive complexity* $K(s)$ of a string s is the length of the *minimal description* of s, i.e., the length of the shortest string encoding $\langle M, w \rangle$ such that the machine M run on input w produces s.

This definition is relatively robust with respect to the type and encoding of our machine M: it differs only by a constant factor.

Theorem

$$\exists c. \forall s.\ K(s) \leq |s| + c$$

Kolmogorov Complexity

Definition

The *descriptive complexity* $K(s)$ of a string s is the length of the *minimal description* of s, i.e., the length of the shortest string encoding $\langle M, w \rangle$ such that the machine M run on input w produces s.

This definition is relatively robust with respect to the type and encoding of our machine M: it differs only by a constant factor.

Theorem

$$\exists c. \forall s.\ K(s) \leq |s| + c$$

Proof: Consider the Turing machine M that immediately halts. Our c can just be the length of $\ulcorner M \urcorner$.

Some Theorems

Given a string s, how much information has ss relative to s?

Some Theorems

Given a string s, how much information has ss relative to s?

Theorem

The string ss has not much more information than s:

$$\exists c. \forall s.\ K(ss) \leq K(s) + c$$

Some Theorems

Given a string s, how much information has ss relative to s?

Theorem

The string ss has not much more information than s:

$$\exists c. \forall s.\ K(ss) \leq K(s) + c$$

Proof: Consider the machine M that takes as input $\langle N, w \rangle$, runs N on w. Once N outputs the string s, M outputs ss.
Let d be the minimal description of s, then a description of ss is $\langle M, d \rangle$, whose length is $K(s) + c$

Returning to Pairing

What's $K(xy)$ for strings x, y?

Returning to Pairing

What's $K(xy)$ for strings x, y? Is it $\leq K(x) + K(y) + c$?

Returning to Pairing

What's $K(xy)$ for strings x, y? Is it $\leq K(x) + K(y) + c$?

No

We can't just concatenate descriptions, as we need to know unambiguously when the description of x ends and the description of y begins.

So: The length of a pair $\langle x, y \rangle$ depends on our pairing method.

Returning to Pairing

What's $K(xy)$ for strings x, y? Is it $\leq K(x) + K(y) + c$?

No

We can't just concatenate descriptions, as we need to know unambiguously when the description of x ends and the description of y begins.

So: The length of a pair $\langle x, y \rangle$ depends on our pairing method.

Sipser's solution

Double every bit in x, and use 01 as a delimiter. Then:

$$K(xy) \leq 2K(x) + K(y) + c$$

Returning to Pairing

What's $K(xy)$ for strings x, y? Is it $\leq K(x) + K(y) + c$?

No

We can't just concatenate descriptions, as we need to know unambiguously when the description of x ends and the description of y begins.

So: The length of a pair $\langle x, y \rangle$ depends on our pairing method.

Sipser's solution

Double every bit in x, and use 01 as a delimiter. Then:

$$K(xy) \leq 2K(x) + K(y) + c$$

By first storing the length of the desc. of x with doubled bits:

$$K(xy) \leq 2\log_2(K(x)) + K(x) + K(y) + c$$

Compressibility

Definition

A string s is *incompressible* if $K(s) \geq |s|$.
Intuitively, these are strings s that can only be described by the program "`print` s".

Compressibility

Definition

A string s is *incompressible* if $K(s) \geq |s|$.
Intuitively, these are strings s that can only be described by the program "`print` s".

Theorem. Incompressible strings of every length exist.

Compressibility

Definition

A string s is *incompressible* if $K(s) \geq |s|$.
Intuitively, these are strings s that can only be described by the program "`print` s".

Theorem. Incompressible strings of every length exist.

Proof

There are 2^n binary strings of length n. The number of descriptions shorter than n is at most:

$$\sum_{i=0}^{n-1} 2^i = 2^n - 1$$

Thus there is at least one string not described by any of these.

Tiny fractions

There is at least one incompressible string of any size.

Tiny fractions

There is at least one incompressible string of any size.

In fact

The vast majority of strings are incompressible.

Tiny fractions

There is at least one incompressible string of any size.

In fact

The vast majority of strings are incompressible.

Theorem

The fraction of strings of size n that are generated by descriptions smaller than $m < n$ is at most 2^{m-n}.

Tiny fractions

There is at least one incompressible string of any size.

In fact

The vast majority of strings are incompressible.

Theorem

The fraction of strings of size n that are generated by descriptions smaller than $m < n$ is at most 2^{m-n}.

Proof: There are 2^n strings of size n, and at most $2^m - 1$ descriptions smaller than m.

Tiny fractions

There is at least one incompressible string of any size.

In fact

The vast majority of strings are incompressible.

Theorem

The fraction of strings of size n that are generated by descriptions smaller than $m < n$ is at most 2^{m-n}.

Proof: There are 2^n strings of size n, and at most $2^m - 1$ descriptions smaller than m. Even if every one of these descriptions produces a string of size n, at most a fraction:

$$\frac{2^m - 1}{2^n} < \frac{2^m}{2^n} = 2^{m-n}$$

of strings of size n can be computed by these descriptions.

Example

Example

Consider a string of eight million bits (roughly 1MB). Assuming all strings are equally likely, what's the probability that this string could be compressed by at least 0.01%?

Example

Example

Consider a string of eight million bits (roughly 1MB). Assuming all strings are equally likely, what's the probability that this string could be compressed by at least 0.01%?

Solution: A compression of 0.01% would mean a description with 7920000 bits, so at most a fraction

$$2^{7920000-8000000} = 2^{-800}$$

of strings of size 1MB can be compressed by 0.01%.

Example

Example

Consider a string of eight million bits (roughly 1MB). Assuming all strings are equally likely, what's the probability that this string could be compressed by at least 0.01%?

Solution: A compression of 0.01% would mean a description with 7920000 bits, so at most a fraction

$$2^{7920000-8000000} = 2^{-800}$$

of strings of size 1MB can be compressed by 0.01%.

Compare

Pick one of the 200 billion galaxies in the observable universe, pick one of its billion stars, pick one of its atoms, then pick one of the protons of that atom. The chances that you and your neighbour guessed the same proton is roughly 2^{-272}.

What's going on?

We have seen in our use of computers that many files we use every day (images, videos, documents) are extremely compressible.

What's going on?

We have seen in our use of computers that many files we use every day (images, videos, documents) are extremely compressible.

Why?

Humans are not interested in **random noise**.

What's going on?

We have seen in our use of computers that many files we use every day (images, videos, documents) are extremely compressible.

Why?

Humans are not interested in **random noise**.

Incompressible strings are also called *random*, and descriptive complexity is proportional to *entropy*.

Similarly, the vast majority of functions are uncomputable, but almost all functions we care about are computable.

Universal Probability

Assume that the complete works of Shakespeare is one million bits long[1]. The probability that a monkey typing at a typewriter produces the complete works of Shakespeare is about:

$$p_{\text{typewriter}} \approx 2^{-1000000}$$

[1] This is a massive underestimate. It is actually around 5MB

Universal Probability

Assume that the complete works of Shakespeare is one million bits long[1]. The probability that a monkey typing at a typewriter produces the complete works of Shakespeare is about:

$$p_{\text{typewriter}} \approx 2^{-1000000}$$

If the monkey is at a computer, however, we only need it to input a program that will produce the works of Shakespeare:

$$p_{\text{computer}} \approx K(\text{Shakespeare})$$

[1] This is a massive underestimate. It is actually around 5MB

Universal Probability

Assume that the complete works of Shakespeare is one million bits long[1]. The probability that a monkey typing at a typewriter produces the complete works of Shakespeare is about:

$$p_{\text{typewriter}} \approx 2^{-1000000}$$

If the monkey is at a computer, however, we only need it to input a program that will produce the works of Shakespeare:

$$p_{\text{computer}} \approx K(\text{Shakespeare})$$

Suppose that $K(\text{Shakespeare}) = 250000$ bits, then the monkey is 2^{750000} times more likely to produce the works of Shakespeare on a computer!

Upshot: Random input is more interesting to a computer than to a typewriter.

[1] This is a massive underestimate. It is actually around 5MB

Berry's Paradox

Is the set of incompressible strings decidable?

Berry's Paradox

Is the set of incompressible strings decidable?

Theorem

No. Assume that it is decidable. Then we could write a machine M that, given a number n as input computes:

for $s \in \{0,1\}^n$:
  if IsIncompressible(s) **then**
    output s; **halt**

Berry's Paradox

Is the set of incompressible strings decidable?

Theorem

No. Assume that it is decidable. Then we could write a machine M that, given a number n as input computes:

for $s \in \{0,1\}^n$:
  if IsIncompressible(s) **then**
    output s; **halt**

Now $\langle M, n \rangle$ is a description of an incompressible string of size n, but the length of $\langle M, n \rangle$ is just $\ulcorner M \urcorner$ (a constant) $+ \log_2 n$.

Berry's Paradox

Is the set of incompressible strings decidable?

Theorem

No. Assume that it is decidable. Then we could write a machine M that, given a number n as input computes:

for $s \in \{0,1\}^n$:
  if IsIncompressible(s) **then**
    output s; **halt**

Now $\langle M, n \rangle$ is a description of an incompressible string of size n, but the length of $\langle M, n \rangle$ is just $\ulcorner M \urcorner$ (a constant) $+ \log_2 n$. We have a paradox! Thus the set of incompressible strings is not decidable.

Berry's Paradox

Is the function K computable?

Berry's Paradox

Is the function K computable?

Theorem

No, by the same reasoning. Assume that K is computable. Then we could write a machine M that, given w, computes:

for $i \in \mathbb{N}$:
  for $s \in \{0,1\}^i$:
    if $K(s) > |\langle M, w \rangle|$ **then**
      output s; **halt**

In English, this is essentially:

"Output the shortest string which can only be described by programs bigger than this one".

Berry's Paradox

Is the function K computable?

Theorem

No, by the same reasoning. Assume that K is computable. Then we could write a machine M that, given w, computes:

for $i \in \mathbb{N}$:
  for $s \in \{0,1\}^i$:
    if $K(s) > |\langle M, w \rangle|$ **then**
      output s; **halt**

In English, this is essentially:

"Output the shortest string which can only be described by programs bigger than this one".

This is a paradox! Thus K is not computable.

We could also do a proof by reduction from the previous incompressibility problem.

One last Theorem

Theorem

Any computably enumerable set of incompressible strings is finite.

One last Theorem

Theorem

Any computably enumerable set of incompressible strings is finite.

Proof: Let $I = \{x \mid K(x) \geq |x|\}$. Assume that S is a computably enumerable infinite subset of I.

One last Theorem

Theorem

Any computably enumerable set of incompressible strings is finite.

Proof: Let $I = \{x \mid K(x) \geq |x|\}$. Assume that S is a computably enumerable infinite subset of I.

Define $h(n) =$ first enumerated string in S of length $\geq n$
Then, h is computable by a machine M.

One last Theorem

Theorem

Any computably enumerable set of incompressible strings is finite.

Proof: Let $I = \{x \mid K(x) \geq |x|\}$. Assume that S is a computably enumerable infinite subset of I.

Define $h(n) =$ first enumerated string in S of length $\geq n$
Then, h is computable by a machine M.

We know:

- $K(h(n)) \geq |h(n)| \geq n$ by the definition of I.
- $K(h(n)) \leq |\langle M, n \rangle| \leq \log_2 n + c$

One last Theorem

Theorem

Any computably enumerable set of incompressible strings is finite.

Proof: Let $I = \{x \mid K(x) \geq |x|\}$. Assume that S is a computably enumerable infinite subset of I.

Define $h(n) =$ first enumerated string in S of length $\geq n$
Then, h is computable by a machine M.

We know:

- $K(h(n)) \geq |h(n)| \geq n$ by the definition of I.
- $K(h(n)) \leq |\langle M, n\rangle| \leq \log_2 n + c$

This is a contradiction as $n > \log_2 n + c$ for large enough n.