Approximating Kolmogorov complexity function

Ruslan Ishkuvatov
LIRMM, CNRS & University of Montpellier
Joint work with Daniil Musatov

Computability in Europe, 2019

Kolmogorov complexity

Kolmogorov complexity

- Kolmogov complexity of a string x: the minimal length of a program (without input) producing x

Kolmogorov complexity

- Kolmogov complexity of a string x: the minimal length of a program (without input) producing x
- $C_U(x) = \min\{|p| : U(p) = x\}$: U is the interpreter, p is the program

Kolmogorov complexity

- Kolmogov complexity of a string x: the minimal length of a program (without input) producing x
- $C_U(x) = \min\{|p| : U(p) = x\}$: U is the interpreter, p is the program
- depends on U

Kolmogorov complexity

- Kolmogov complexity of a string x: the minimal length of a program (without input) producing x
- $C_U(x) = \min\{|p| : U(p) = x\}$: U is the interpreter, p is the program
- depends on U
- there exists optimal U that makes C_U minimal up to $O(1)$

Kolmogorov complexity

- Kolmogov complexity of a string x: the minimal length of a program (without input) producing x
- $C_U(x) = \min\{|p| : U(p) = x\}$: U is the interpreter, p is the program
- depends on U
- there exists optimal U that makes C_U minimal up to $O(1)$

$$\exists U\, \forall U'\, \exists c\, \forall x\, [C_U(x) \leqslant C_{U'}(x) + c]$$

Kolmogorov complexity

- Kolmogov complexity of a string x: the minimal length of a program (without input) producing x
- $C_U(x) = \min\{|p| : U(p) = x\}$: U is the interpreter, p is the program
- depends on U
- there exists optimal U that makes C_U minimal up to $O(1)$

$$\exists U \, \forall U' \, \exists c \, \forall x \, [C_U(x) \leqslant C_{U'}(x) + c]$$

- fix some U and forget about it

Non-computability

Non-computability

- Kolmogorov complexity is not computable

Non-computability

- Kolmogorov complexity is not computable
- Berry – Chaitin argument:
  «the first string of complexity more than $10^{10^{10}}$»

Non-computability

- Kolmogorov complexity is not computable
- Berry – Chaitin argument:
  «the first string of complexity more than $10^{10^{10}}$»
- moreover, every computable lower bound is trivial, even if partial

Non-computability

- Kolmogorov complexity is not computable
- Berry – Chaitin argument:
  «the first string of complexity more than $10^{10^{10}}$»
- moreover, every computable lower bound is trivial, even if partial
- Kolmogorov complexity is not generically computable

Non-computability

- Kolmogorov complexity is not computable
- Berry – Chaitin argument:
  «the first string of complexity more than $10^{10^{10}}$»
- moreover, every computable lower bound is trivial, even if partial
- Kolmogorov complexity is not generically computable
- coarse computability: a total algorithm that makes few errors

Non-computability

- Kolmogorov complexity is not computable
- Berry – Chaitin argument:
  «the first string of complexity more than $10^{10^{10}}$»
- moreover, every computable lower bound is trivial, even if partial
- Kolmogorov complexity is not generically computable
- coarse computability: a total algorithm that makes few errors
- total function $f(\cdot)$ is *coarsely computable* if there is a total computable F such that the fraction of errors converges to 0:

$$\frac{\#\{i\colon i < N \text{ and } f(i) \neq F(i)\}}{N} \to 0 \quad \text{as } N \to \infty$$

Approximation errors

Approximation errors

- Theorem: Kolmogorov complexity function is not coarsely computable

Approximation errors

- Theorem: Kolmogorov complexity function is not coarsely computable
- $C^t(x)$ is the minimal length of a program that produces x in time at most t

Approximation errors

- Theorem: Kolmogorov complexity function is not coarsely computable
- $C^t(x)$ is the minimal length of a program that produces x in time at most t
- a computable upper bound for $C\ldots$

Approximation errors

- Theorem: Kolmogorov complexity function is not coarsely computable
- $C^t(x)$ is the minimal length of a program that produces x in time at most t
- a computable upper bound for C...
- ...therefore the fraction of errors does not converge to 0

Approximation errors

- Theorem: Kolmogorov complexity function is not coarsely computable
- $C^t(x)$ is the minimal length of a program that produces x in time at most t
- a computable upper bound for C...
- ...therefore the fraction of errors does not converge to 0
- improvements:

Approximation errors

- Theorem: Kolmogorov complexity function is not coarsely computable
- $C^t(x)$ is the minimal length of a program that produces x in time at most t
- a computable upper bound for C...
- ...therefore the fraction of errors does not converge to 0
- improvements: ...is separated from 0

Approximation errors

- Theorem: Kolmogorov complexity function is not coarsely computable
- $C^t(x)$ is the minimal length of a program that produces x in time at most t
- a computable upper bound for $C\ldots$
- $\ldots$therefore the fraction of errors does not converge to 0
- improvements: $\ldots$is separated from 0
- what if we approximate complexity values?

Approximation errors

- Theorem: Kolmogorov complexity function is not coarsely computable
- $C^t(x)$ is the minimal length of a program that produces x in time at most t
- a computable upper bound for C...
- ...therefore the fraction of errors does not converge to 0
- improvements: ...is separated from 0
- what if we approximate complexity values?
- difficulty in the Medvedev lattice

Approximation errors

- Theorem: Kolmogorov complexity function is not coarsely computable
- $C^t(x)$ is the minimal length of a program that produces x in time at most t
- a computable upper bound for C...
- ...therefore the fraction of errors does not converge to 0
- improvements: ...is separated from 0
- what if we approximate complexity values?
- difficulty in the Medvedev lattice
- logical implications

Trade-off: precision vs #errors

Trade-off: precision vs #errors

- length is an upper bound: $C(x) \leq |x| + O(1)$

Trade-off: precision vs #errors

- length is an upper bound: $C(x) \leq |x| + O(1)$
- $C(x) < |x| - d$: compressible (by d) strings

Trade-off: precision vs #errors

- length is an upper bound: $C(x) \leq |x| + O(1)$
- $C(x) < |x| - d$: compressible (by d) strings are rare (fraction: $\approx 2^{-d}$)

Trade-off: precision vs #errors

- length is an upper bound: $C(x) \leq |x| + O(1)$
- $C(x) < |x| - d$: compressible (by d) strings are rare (fraction: $\approx 2^{-d}$)
- increasing d: lower precision required, less errors

Trade-off: precision vs #errors

- length is an upper bound: $C(x) \leq |x| + O(1)$
- $C(x) < |x| - d$: compressible (by d) strings are rare (fraction: $\approx 2^{-d}$)
- increasing d: lower precision required, less errors
- «Theorem»: No approximation is better than length: the same trade-off

Trade-off: precision vs #errors

- length is an upper bound: $C(x) \leq |x| + O(1)$
- $C(x) < |x| - d$: compressible (by d) strings are rare (fraction: $\approx 2^{-d}$)
- increasing d: lower precision required, less errors
- «Theorem»: No approximation is better than length: the same trade-off
- technical: one-sided error d is two-sided error $d/2$: take $C(x) - d/2$

Trade-off: precision vs #errors

- length is an upper bound: $C(x) \leq |x| + O(1)$
- $C(x) < |x| - d$: compressible (by d) strings are rare (fraction: $\approx 2^{-d}$)
- increasing d: lower precision required, less errors
- «Theorem»: No approximation is better than length: the same trade-off
- technical: one-sided error d is two-sided error $d/2$: take $C(x) - d/2$
- Let $e(n)$ and $d(n)$ be two total computable functions with integer values, $e(n) - 2d(n) \to \infty$ and $e(n) \leq n$. Then there is no computable function $\tilde{C}(x)$ that is $(d(n), e(n))$-approximation for $C(x)$ on n-bit strings.

Trade-off: precision vs #errors

- length is an upper bound: $C(x) \leq |x| + O(1)$
- $C(x) < |x| - d$: compressible (by d) strings are rare (fraction: $\approx 2^{-d}$)
- increasing d: lower precision required, less errors
- «Theorem»: No approximation is better than length: the same trade-off
- technical: one-sided error d is two-sided error $d/2$: take $C(x) - d/2$
- Let $e(n)$ and $d(n)$ be two total computable functions with integer values, $e(n) - 2d(n) \to \infty$ and $e(n) \leq n$. Then there is no computable function $\tilde{C}(x)$ that is $(d(n), e(n))$-approximation for $C(x)$ on n-bit strings.
- (d, e)-approximation: deviation $< d$ except for 2^{-e}-fraction

Finite version and proof idea

Finite version and proof idea

- looking for a program that approximates complexity for n-bit strings

Finite version and proof idea

- looking for a program that approximates complexity for n-bit strings
- exists (just a table)

Finite version and proof idea

- looking for a program that approximates complexity for n-bit strings
- exists (just a table)
- . . . but has high complexity

Finite version and proof idea

- looking for a program that approximates complexity for n-bit strings
- exists (just a table)
- ... but has high complexity
- (d, e)-approximation for n-bit strings requires complexity about $e - 2d$

Finite version and proof idea

- looking for a program that approximates complexity for n-bit strings
- exists (just a table)
- ... but has high complexity
- (d, e)-approximation for n-bit strings requires complexity about $e - 2d$
- why? knowing (a) the approximation; (b) number of places where approximation has large error [$n - e$ bits] we can find those places and then find a string of complexity at least $n - 2d$, so

$$\text{(complexity of the approximation)} + (n - e) \geq n - 2d$$

Finite version and proof idea

- looking for a program that approximates complexity for n-bit strings
- exists (just a table)
- …but has high complexity
- (d, e)-approximation for n-bit strings requires complexity about $e - 2d$
- why? knowing (a) the approximation; (b) number of places where approximation has large error [$n - e$ bits] we can find those places and then find a string of complexity at least $n - 2d$, so

$$(\text{complexity of the approximation}) + (n - e) \geq n - 2d$$

- many technical details omitted ("one-sided" errors, logarithmic terms, etc.)

Approximation as a mass problem

Approximation as a mass problem

- mass problem (Medvedev): a set of total functions ("solutions")

Approximation as a mass problem

- mass problem (Medvedev): a set of total functions ("solutions")
- approximating complexity: a set of good approximations

Approximation as a mass problem

- mass problem (Medvedev): a set of total functions ("solutions")
- approximating complexity: a set of good approximations
- Medvedev reducibility $\mathcal{A} \le \mathcal{B}$: there is a machine M such that $M^B \in \mathcal{A}$ for each $B \in \mathcal{B}$

Approximation as a mass problem

- mass problem (Medvedev): a set of total functions ("solutions")
- approximating complexity: a set of good approximations
- Medvedev reducibility $\mathcal{A} \le \mathcal{B}$: there is a machine M such that $M^B \in \mathcal{A}$ for each $B \in \mathcal{B}$
- Theorem: halting problem is Medvedev-reducible to the problem of approximating complexity (with any given non-trivial precision)

Approximation as a mass problem

- mass problem (Medvedev): a set of total functions ("solutions")
- approximating complexity: a set of good approximations
- Medvedev reducibility $\mathcal{A} \le \mathcal{B}$: there is a machine M such that $M^B \in \mathcal{A}$ for each $B \in \mathcal{B}$
- Theorem: halting problem is Medvedev-reducible to the problem of approximating complexity (with any given non-trivial precision)
- even separating incompressible strings from 100-fold compressible is above the halting problem

Approximation and logic

Approximation and logic

- for every string x and integer n: arithmetical statement $C(x) \geq n$

Approximation and logic

- for every string x and integer n: arithmetical statement $C(x) \geq n$
- they are Π_1-statements

Approximation and logic

- for every string x and integer n: arithmetical statement $C(x) \geq n$
- they are Π_1-statements
- a lot of true statements (most of n-bit strings are almost incompressible)

Approximation and logic

- for every string x and integer n: arithmetical statement $C(x) \geq n$
- they are Π_1-statements
- a lot of true statements (most of n-bit strings are almost incompressible)
- none of them provable (except for bounded values of n)

Approximation and logic

- for every string x and integer n: arithmetical statement $C(x) \geq n$
- they are Π_1-statements
- a lot of true statements (most of n-bit strings are almost incompressible)
- none of them provable (except for bounded values of n)
- what if we add them as axioms?

Approximation and logic

- for every string x and integer n: arithmetical statement $C(x) \geq n$
- they are Π_1-statements
- a lot of true statements (most of n-bit strings are almost incompressible)
- none of them provable (except for bounded values of n)
- what if we add them as axioms?
- Theorem: adding axioms $C(x) \geq |x|/2$ for every incompressible string x, we may derive all true universal statements

Approximation and logic

- for every string x and integer n: arithmetical statement $C(x) \geq n$
- they are Π_1-statements
- a lot of true statements (most of n-bit strings are almost incompressible)
- none of them provable (except for bounded values of n)
- what if we add them as axioms?
- Theorem: adding axioms $C(x) \geq |x|/2$ for every incompressible string x, we may derive all true universal statements
- (a logic counterpart of the difficulty of separation)

Thanks!