Kolmogorov complexity: a primer

Alexander Shen,
LIF CNRS & Univ. Aix – Marseille

June 26, 2009

Information "density"

- The same English text uses four times more space in UTF-32 compared to ASCII

Information "density"

- The same English text uses four times more space in UTF-32 compared to ASCII
- $$\text{density} = \frac{\text{weight}}{\text{volume}}$$

Information "density"

- The same English text uses four times more space in UTF-32 compared to ASCII
- $$\text{density} = \frac{\text{weight}}{\text{volume}}$$
- $$\text{information density} = \frac{\text{amount of information}}{\text{length}}$$

Amount of information

- Kolmogorov, 1965: “Three approaches to the definition of the amount of information”

Amount of information

- Kolmogorov, 1965: "Three approaches to the definition of the amount of information"
- Shannon approach: entropy of a random variable

Amount of information

- Kolmogorov, 1965: "Three approaches to the definition of the amount of information"
- Shannon approach: entropy of a random variable
- 0.1 : 0.9 random variable: about 0.47 bits/value
- 0.5 : 0.5 random variable: 1 bit/value

Amount of information

- Kolmogorov, 1965: "Three approaches to the definition of the amount of information"
- Shannon approach: entropy of a random variable
- 0.1 : 0.9 random variable: about 0.47 bits/value
- 0.5 : 0.5 random variable: 1 bit/value
- Not usable for finite object (e.g., DNA, files, etc.)

Compressed size

Compressed size

- Original file F contains f bits

Compressed size

- Original file F contains f bits
- Compressed file $G = C(F)$ contains g bits

Compressed size

- Original file F contains f bits
- Compressed file $G = C(F)$ contains g bits (we hope that $g < f$)

Compressed size

- Original file F contains f bits
- Compressed file $G = C(F)$ contains g bits (we hope that $g < f$)
- Decompressing produces original file:

$$F = D(G) = D(C(F))$$

Compressed size

- Original file F contains f bits
- Compressed file $G = C(F)$ contains g bits (we hope that $g < f$)
- Decompressing produces original file:

$$F = D(G) = D(C(F))$$

- `gzip`, `gunzip`

Choosing a (de)compressor

Choosing a (de)compressor

- How good the compression could be?

Choosing a (de)compressor

- How good the compression could be? Is it possible that every string (of length, say, 100000) is compressed by at least 10%?

Choosing a (de)compressor

- How good the compression could be? Is it possible that every string (of length, say, 100000) is compressed by at least 10%?
- Compressor should be injective: $X \neq Y \Rightarrow C(X) \neq C(Y)$

Choosing a (de)compressor

- How good the compression could be? Is it possible that every string (of length, say, 100000) is compressed by at least 10%?
- Compressor should be injective: $X \neq Y \Rightarrow C(X) \neq C(Y)$
- Trade-off: for any string there is a compressor/decompressor pair that compresses this string well

Comparing (de)compressors

Comparing (de)compressors

- C_1/D_1 is better than C_2/D_2 if $|C_1(F)| \leq |C_2(F)|$ for every F

Comparing (de)compressors

- C_1/D_1 is better than C_2/D_2 if $|C_1(F)| \leq |C_2(F)|$ for every F
- C_1/D_1 is *asymptotically* better than C_2/D_2 if $|C_1(F)| \leq |C_2(F)| + c$ for every F and some c (not depending on F)

Comparing (de)compressors

- C_1/D_1 is better than C_2/D_2 if $|C_1(F)| \leq |C_2(F)|$ for every F
- C_1/D_1 is *asymptotically* better than C_2/D_2 if $|C_1(F)| \leq |C_2(F)| + c$ for every F and some c (not depending on F)
- Combining two compressors: for every two C_1/D_1 and C_2/D_2 there exists C/D asymptotically better than both

Comparing (de)compressors

- C_1/D_1 is better than C_2/D_2 if $|C_1(F)| \leq |C_2(F)|$ for every F
- C_1/D_1 is *asymptotically* better than C_2/D_2 if $|C_1(F)| \leq |C_2(F)| + c$ for every F and some c (not depending on F)
- Combining two compressors: for every two C_1/D_1 and C_2/D_2 there exists C/D asymptotically better than both
- Proof: use C_1 or C_2 (whichever is better) and use the first bit to record this choice.

Comparing (de)compressors

- C_1/D_1 is better than C_2/D_2 if $|C_1(F)| \leq |C_2(F)|$ for every F
- C_1/D_1 is *asymptotically* better than C_2/D_2 if $|C_1(F)| \leq |C_2(F)| + c$ for every F and some c (not depending on F)
- Combining two compressors: for every two C_1/D_1 and C_2/D_2 there exists C/D asymptotically better than both
- Proof: use C_1 or C_2 (whichever is better) and use the first bit to record this choice.
- Does an optimal compressor/decompressor pair exist?

No optimal compressor

No optimal compressor

- For every C/D there is C'/D' that is much better than C/D on some strings.

No optimal compressor

- For every C/D there is C'/D' that is much better than C/D on some strings. Why?

No optimal compressor

- For every C/D there is C'/D' that is much better than C/D on some strings. Why?
- For any length n there exists C-*incompressible* string of length n.

No optimal compressor

- For every C/D there is C'/D' that is much better than C/D on some strings. Why?
- For any length n there exists C-*incompressible* string of length n. (Pigeon-hole principle)

No optimal compressor

- For every C/D there is C'/D' that is much better than C/D on some strings. Why?
- For any length n there exists C*-incompressible* string of length n. (Pigeon-hole principle)
- New C' compresses the first C-incompressible string of length n into 0binary(n) and any other x of length n into $1x$.

Kolmogorov complexity

Kolmogorov complexity

- No compressors

Kolmogorov complexity

- No compressors
- Decompressor = any *partial* computable function (=algorithm) that maps strings to strings

Kolmogorov complexity

- No compressors
- Decompressor = any *partial* computable function (=algorithm) that maps strings to strings
- $K_D(x) = \{\min |p| \colon D(p) = x\}$

Kolmogorov complexity

- No compressors
- Decompressor = any *partial* computable function (=algorithm) that maps strings to strings
- $K_D(x) = \{\min |p| \colon D(p) = x\}$
- D_1 is (asymptotically) better that D_2 if $K_{D_1}(x) \leq K_{D_2}(x) + c$ for all x and some c.

Kolmogorov complexity

- No compressors
- Decompressor = any *partial* computable function (=algorithm) that maps strings to strings
- $K_D(x) = \{\min |p| : D(p) = x\}$
- D_1 is (asymptotically) better that D_2 if $K_{D_1}(x) \leq K_{D_2}(x) + c$ for all x and some c.
- There exist an optimal decompressor (=asymptotically better than any other one)

Optimal decompressor exists

Optimal decompressor exists

- Idea: self-extracting archive

Optimal decompressor exists

- Idea: self-extracting archive
- Decompressor:

  • Read from left to right until a self-delimited program is found

  • Run this program on the rest of input

Optimal decompressor exists

- Idea: self-extracting archive
- Decompressor:

  • Read from left to right until a self-delimited program is found

  • Run this program on the rest of input
- Universal interpreter U

Optimal decompressor exists

- Idea: self-extracting archive
- Decompressor:

  • Read from left to right until a self-delimited program is found

  • Run this program on the rest of input
- Universal interpreter U
- $K_U(x) \leq K_D(x) +$ the length of D program

Is Kolmogorov complexity well defined?

Is Kolmogorov complexity well defined?

- Fix some optimal decompressor D

Is Kolmogorov complexity well defined?

- Fix some optimal decompressor D
- Call $K_D(x)$ the *Kolmogorov complexity* of D

Is Kolmogorov complexity well defined?

- Fix some optimal decompressor D
- Call $K_D(x)$ the *Kolmogorov complexity* of D
- $K(x)$ is not really defined for an individual x, but

Is Kolmogorov complexity well defined?

- Fix some optimal decompressor D
- Call $K_D(x)$ the *Kolmogorov complexity* of D
- $K(x)$ is not really defined for an individual x, but
- function K is defined up to a $O(1)$ additive term

Is Kolmogorov complexity well defined?

- Fix some optimal decompressor D
- Call $K_D(x)$ the *Kolmogorov complexity* of D
- $K(x)$ is not really defined for an individual x, but
- function K is defined up to a $O(1)$ additive term
- "natural" choices of D lead to K_D that differ not very much

Berry paradox and non-computability

Berry paradox and non-computability

- D': $n \mapsto$ the first string of complexity $> n$

Berry paradox and non-computability

- D': $n \mapsto$ the first string of complexity $> n$
- D'-complexity of this string is about $\log n$: a contradiction?

Berry paradox and non-computability

- $D'\colon n \mapsto$ the first string of complexity $> n$
- D'-complexity of this string is about $\log n$: a contradiction?
- Theorem: complexity function is not computable

Berry paradox and non-computability

- $D' \colon n \mapsto$ the first string of complexity $> n$
- D'-complexity of this string is about $\log n$: a contradiction?
- Theorem: complexity function is not computable
- Theorem: there is no algorithmic way to generate strings of high complexity.

- Most strings of given length are incompressible

- Most strings of given length are incompressible
- Random string is incompressible with high probability

- Most strings of given length are incompressible
- Random string is incompressible with high probability

  A lot of information, but not very useful

Information in X about Y

Information in X about Y

- Conditional complexity of Y when X is known: $K_D(Y|X) = \min\{l(P) \colon D(P, X) = Y\}$

Information in X about Y

- Conditional complexity of Y when X is known: $K_D(Y|X) = \min\{l(P)\colon D(P, X) = Y\}$
- Optimal conditional decompressor exists; conditional Kolmogorov complexity $K(Y|X)$

Information in X about Y

- Conditional complexity of Y when X is known: $K_D(Y|X) = \min\{l(P)\colon D(P,X) = Y\}$
- Optimal conditional decompressor exists; conditional Kolmogorov complexity $K(Y|X)$
- $K(X|X) \approx 0$

Information in X about Y

- Conditional complexity of Y when X is known: $K_D(Y|X) = \min\{I(P) \colon D(P, X) = Y\}$
- Optimal conditional decompressor exists; conditional Kolmogorov complexity $K(Y|X)$
- $K(X|X) \approx 0$
- $K(XX|X) \approx 0$

Information in X about Y

- Conditional complexity of Y when X is known: $K_D(Y|X) = \min\{l(P) \colon D(P, X) = Y\}$
- Optimal conditional decompressor exists; conditional Kolmogorov complexity $K(Y|X)$
- $K(X|X) \approx 0$
- $K(XX|X) \approx 0$
- $K(Y|X) \leq K(Y) + O(1)$

Information in X about Y

- Conditional complexity of Y when X is known: $K_D(Y|X) = \min\{l(P) \colon D(P,X) = Y\}$
- Optimal conditional decompressor exists; conditional Kolmogorov complexity $K(Y|X)$
- $K(X|X) \approx 0$
- $K(XX|X) \approx 0$
- $K(Y|X) \leq K(Y) + O(1)$
- $I(X:Y) = K(Y) - K(Y|X)$

Information in X about Y

- Conditional complexity of Y when X is known: $K_D(Y|X) = \min\{l(P)\colon D(P,X) = Y\}$
- Optimal conditional decompressor exists; conditional Kolmogorov complexity $K(Y|X)$
- $K(X|X) \approx 0$
- $K(XX|X) \approx 0$
- $K(Y|X) \leq K(Y) + O(1)$
- $I(X:Y) = K(Y) - K(Y|X)$
- $I(X:Y) = I(Y:X)$

Randomness=incompressibility

Randomness=incompressibility

- Why we don't belive in a fair coin that produces $0101010101\ldots01$?

Randomness=incompressibility

- Why we don't belive in a fair coin that produces $0101010101\ldots01$?
- Random string = incompressible string

Randomness=incompressibility

- Why we don't belive in a fair coin that produces $0101010101\ldots01$?
- Random string = incompressible string
- Theorem: random string has frequency of ones about 50%

Randomness=incompressibility

- Why we don't belive in a fair coin that produces $0101010101\ldots01$?
- Random string = incompressible string
- Theorem: random string has frequency of ones about 50%
- Other laws of probability theory (e.g., frequencies of groups)

Probabilistic proofs

Probabilistic proofs

- To prove existence of something: show that this event has positive probability

Probabilistic proofs

- To prove existence of something: show that this event has positive probability
- Example: bit string of length 1000 that is square-free for substrings of size 100

Probabilistic proofs

- To prove existence of something: show that this event has positive probability
- Example: bit string of length 1000 that is square-free for substrings of size 100
- Complexity version: prove that incompressible string of length 1000 is square free.

Probabilistic proofs

- To prove existence of something: show that this event has positive probability
- Example: bit string of length 1000 that is square-free for substrings of size 100
- Complexity version: prove that incompressible string of length 1000 is square free. Large square means compressibility

Best paper of STOC'2009 (Robin Moser)

Best paper of STOC'2009 (Robin Moser)

- N bit variables $b_1, \ldots, b_N$

Best paper of STOC'2009 (Robin Moser)

- N bit variables $b_1, \ldots, b_N$
- some number of clauses of size m:

$$((\neg)b_{i_1} \vee (\neg)b_{i_2} \vee \ldots \vee (\neg)b_{i_m})$$

Best paper of STOC'2009 (Robin Moser)

- N bit variables $b_1, \ldots, b_N$
- some number of clauses of size m:

$$((\neg)b_{i_1} \vee (\neg)b_{i_2} \vee \ldots \vee (\neg)b_{i_m})$$

- Neighbor clauses: clauses that share some variable

Best paper of STOC'2009 (Robin Moser)

- N bit variables $b_1, \ldots, b_N$
- some number of clauses of size m:

$$((\neg)b_{i_1} \vee (\neg)b_{i_2} \vee \ldots \vee (\neg)b_{i_m})$$

- Neighbor clauses: clauses that share some variable
- Each clause has at most r neighbors; $r = o(2^m)$

Best paper of STOC'2009 (Robin Moser)

- N bit variables $b_1, \ldots, b_N$
- some number of clauses of size m:

$$((\neg)b_{i_1} \vee (\neg)b_{i_2} \vee \ldots \vee (\neg)b_{i_m})$$

- Neighbor clauses: clauses that share some variable
- Each clause has at most r neighbors; $r = o(2^m)$
- Then there exists a satisfying assignment

Proof inspired by Kolmogorov complexity

Proof inspired by Kolmogorov complexity

- Start with an incompressible source of random bits

Proof inspired by Kolmogorov complexity

- Start with an incompressible source of random bits
- Use them to initialize variables

Proof inspired by Kolmogorov complexity

- Start with an incompressible source of random bits
- Use them to initialize variables
- If all clauses are satisfied, then OK

Proof inspired by Kolmogorov complexity

- Start with an incompressible source of random bits
- Use them to initialize variables
- If all clauses are satisfied, then OK
- If not, “fix” all of them sequentially

Proof inspired by Kolmogorov complexity

- Start with an incompressible source of random bits
- Use them to initialize variables
- If all clauses are satisfied, then OK
- If not, “fix” all of them sequentially
- Fix a clause: reinitialize it with fresh random bits until it is true; fix neighbor clauses that are damaged

Proof inspired by Kolmogorov complexity

- Start with an incompressible source of random bits
- Use them to initialize variables
- If all clauses are satisfied, then OK
- If not, "fix" all of them sequentially
- Fix a clause: reinitialize it with fresh random bits until it is true; fix neighbor clauses that are damaged
- If this does not terminate for a long time, the random source is compressible (it is determined by the list of corrected clauses)

Other applications

Other applications

- Definition of an infinite random sequence

Other applications

- Definition of an infinite random sequence
- Connection with recursion theory

Other applications

- Definition of an infinite random sequence
- Connection with recursion theory
- Shannon information theory and algorithmic information theory

Other applications

- Definition of an infinite random sequence
- Connection with recursion theory
- Shannon information theory and algorithmic information theory
- Combinatorial interpretation. Example: $V^2 \leq S_1 S_2 S_3$