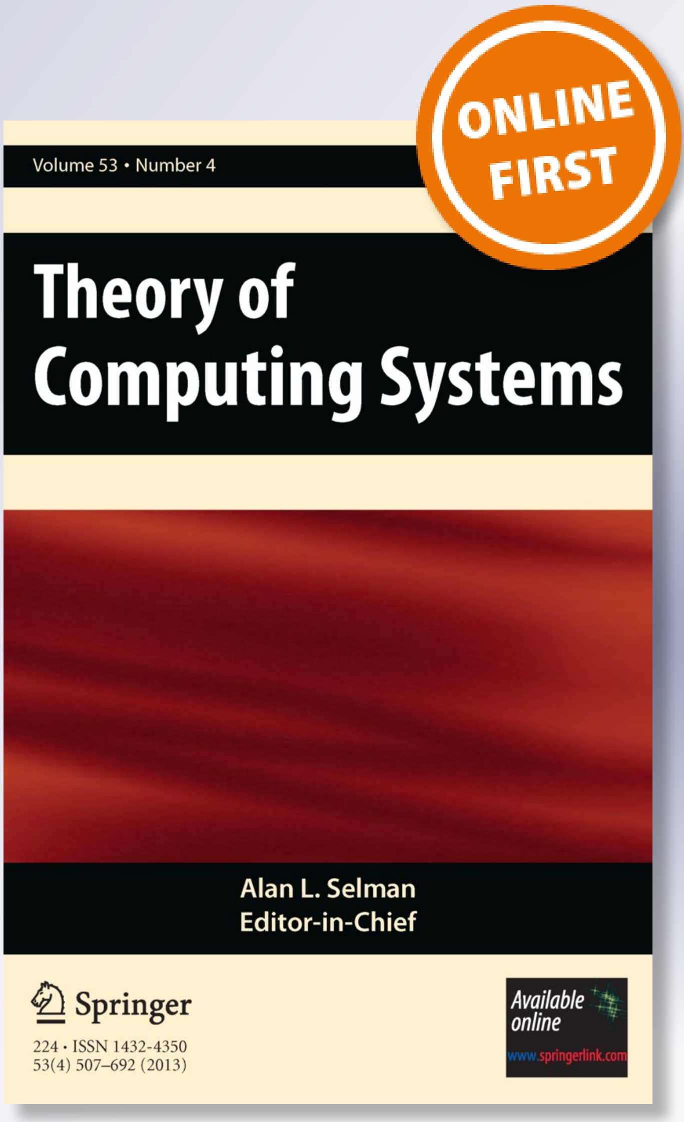
On Algorithmic Statistics for Space-bounded Algorithms

Alexey Milovanov

Theory of Computing Systems

ISSN 1432-4350

Theory Comput Syst
DOI 10.1007/s00224-018-9845-6



Theory Comput Syst
https://doi.org/10.1007/s00224-018-9845-6

On Algorithmic Statistics for Space-bounded Algorithms

Alexey Milovanov[1]

Abstract Algorithmic statistics looks for models of observed data that are good in the following sense: a model is simple (i.e., has small Kolmogorov complexity) and captures all the algorithmically discoverable regularities in the data. However, this idea can not be used in practice as is because Kolmogorov complexity is not computable. In this paper we develop an algorithmic version of algorithmic statistics that uses space-bounded Kolmogorov complexity. We prove a space-bounded version of a basic result from "classical" algorithmic statistics, the connection between optimality and randomness deficiences. The main tool is the Nisan–Wigderson pseudo-random generator. An extended abstract of this paper was presented at the 12th International Computer Science Symposium in Russia (Milovanov 2017).

Keywords Algorithmic statistics · Kolmogorov complexity · Nisan-Wigderson generator · Computational complexity theory · Derandomization

This article is part of the Topical Collection on *Computer Science Symposium in Russia*

✉ Alexey Milovanov
almas239@gmail.com

1 National Research University Higher School of Economics, Moscow, Russia

Published online: 06 February 2018

1 Algorithmic Statistics: a Reminder

We consider strings over the binary alphabet {0, 1}. We use $|x|$ to denote the length of a string x. All logarithms are binary. The conditional Kolmogorov complexity[1] of x given y is denoted by $\mathrm{C}(x|y)$.

Let x be some observation data encoded as a binary string. We look for a suitable explanation for x. An explanation (= model) is a finite set containing x. More specifically, we want to find a simple model A such that x is a typical element in A. First of all we need to define what are simple models and typical elements. In classical algorithmic statistics a set A is called simple if its Kolmogorov complexity $\mathrm{C}(A)$ is small. [2] To define the notion of a typical element, we use the *randomness deficiency of x as an element of A*:

$$d(x|A) := \log|A| - \mathrm{C}(x|A).$$

The randomness deficiency is always non-negative (with $O(1)$ accuracy; note that complexity is defined up to $O(1)$ additive term). Indeed, we can find x given A and the index of x in A. For each set A, most elements x of A have small randomness deficiency $d(x|A)$: the fraction of x in A with randomness deficiency greater than β is less than $2^{-\beta}$.

There is another quantity measuring the quality of A as an explanation of x: the *optimality deficiency*

$$\delta(x, A) := \mathrm{C}(A) + \log|A| - \mathrm{C}(x).$$

It is non-negative with logarithmic accuracy (for similar reasons). This quantity represents the following idea: a good explanation (a set) should be simple but also should be small, so the "two-part" description of x (first we specify A, then the ordinal number of x in A) is almost optimal in terms of length.

One could wonder why we consider only sets as explanations and not general probability distributions (in other terms, why we restrict ourselves to uniform probability distributions). Indeed, one can define the notions of complexity and deficiency for distributions on binary strings with finite support and rational values.[3] Still this extension is not essential: for every string x and for every distribution P there exists a set $A \ni x$ explaining x that is almost as good as P, as the following observation [19] shows:

Proposition 1 *For every string x and for every distribution P there exists a set $A \ni x$ such that* $\mathrm{C}(A|P) \le O(\log|x|)$ *and* $\frac{1}{|A|} \ge \frac{1}{2}P(x)$.

[1] The definition and basic properties of Kolmogorov complexity can be found in the textbooks [7, 16]; for a short survey see [14].

[2] Kolmogorov complexity of a finite set A is defined as follows. We fix some computable bijection (*encoding*) $A \mapsto [A]$ from the family of all finite sets to the set of all binary strings. Then we define $\mathrm{C}(A)$ as the complexity $\mathrm{C}([A])$ of the code $[A]$ of A.

[3] The randomness deficiency of a string x with respect to a distribution P is defined as $d(x|P) := -\log P(x) - \mathrm{C}(x|P)$. The optimality deficiency is defined as $\delta(x, P) := \mathrm{C}(P) - \log P(x) - \mathrm{C}(x)$. See [18] for details.

Kolmogorov called a string x *stochastic* if there exists a set $A \ni x$ such that $\mathrm{C}(A) \approx 0$ and $d(x|A) \approx 0$. The last equality means that $\log|A| \approx \mathrm{C}(x|A)$ and therefore $\log|A| \approx \mathrm{C}(x)$ because $\mathrm{C}(A) \approx 0$ (the first condition). So, $\delta(x, A)$ is also small for good explanations.

For example, an incompressible string of length n (i.e., a string whose complexity is close to its length n) is stochastic—the corresponding set is $\{0, 1\}^n$. Non-stochastic objects also exist [15, 19].

2 Space-bounded Complexities

As mentioned by Kolmogorov in [6], the notion of Kolmogorov complexity $\mathrm{C}(x)$ belongs to general computability theory: It ignores time and space needed to produce x from its short description. To take the resources into account, we may introduce space- or time-bounded Kolmogorov complexities (see, for example, [2] or [17]). In this paper we consider space bounds and require that the space used by an algorithm is bounded by a polynomial in the length of its input.

The Kolmogorov complexity of a string x is defined as the minimal length of a program that produces x (on the empty input) and uses at most m bits of memory.[4] We denote this value by $\mathrm{C}^m(x)$ (since we do not consider time bounds, the superscript m always denotes the space bound). If for some x and m such a program p does not exist, we let $\mathrm{C}^m(x) := \infty$.

Of course, this definition depends on the programming language used. We need to fix some "universal" language. More formally, a Turing machine V with two inputs is called *universal* if for every machine U with two inputs and for every q there exists p with $|p| \le |q| + O(1)$ such that $V(p, y) = U(q, y)$ for every y, and V uses space at most $O(m)$ if U uses space m on input (q, y). Here the constant in $O(m)$ depends on U but does not depend on q. Such a universal machine does exist. Indeed, we may let $p = \hat{U}q$ where $\hat{U}$ is a self-delimiting description of U; the machine V first decodes U and then simulates U on the rest of the input p (i.e., q) and the second input. It is important that the simulation overhead in terms of space is only a constant factor. (This is a standard construction used in the definition of the resource-bounded complexity; see, for example [7].)

Now we fix some universal machine V and interpret the statement "program p produces x on the empty input" as $V(p, \Lambda) = x$, where Λ denotes the empty string. The resulting definition depend on the choice of V, but this dependence is limited. Namely, the following proposition holds:

Proposition 2 ([7, Theorem 7.1.1]) *Let V be a universal machine in the sense explained above. Then for every machine U there exists some constants c_1, c_2 such that*

$$\mathrm{C}_V^{c_1 m}(x) \le \mathrm{C}_U^m(x) + c_2$$

[4] We agree that only work tape cells (but not the cells on input or output tapes) are taken into account.

for every m and x, where the subscripts U and V denote the machine used in the definition.

In the sequel we do not mention the universal machine V explicitly and denote $V(p.x)$ as $p(x)$ and call it the output of program p on input x. Similar convention is used for oracle computations (see below).

We need to extend the notion of space-bounded complexity to arbitrary sets (in fact, we will use only finite sets). The (decision) complexity of a set A with space bound m is defined as the minimal length of a program p such that

- $p(y) = 1$ if $y \in A$;
- $p(y) = 0$ if $y \notin A$;
- p uses at most m bits of memory on every input.

Denote this value as $\mathrm{C}^m(A)$. (As before, the universal machine is used in this definition, and Proposition 2 can be easily extended to this case.)

We also need conditional complexities. Both strings and sets (oracles) will be used as conditions. The definitions given above extend naturally to this case; we do not count the cells on the oracle tape when speaking about used memory. For example, the value $\mathrm{C}^m(x|A)$ is defined as the minimal length of a program that on empty input and with oracle A, uses space at most m and produces x. The value $\mathrm{C}^m(B|A)$ for arbitrary sets A, B is defined in a similar way. The value $\mathrm{C}^m(A|x)$ is defined as the minimal length of a program p such that $p(y, x)$ equals 1 for $y \in A$ and 0 for $y \notin A$, and uses space at most m. (Here we need to consider universal machines with three inputs instead of two, but it can be done in a straightforward way.)

For space-bounded complexity the complexity of a string is close to the complexity of a singleton that consists of this string:

Proposition 3 *There exists constants c_1 and c_2 such that*

$$\mathrm{C}^{c_1(m+|x|)}(x) \le \mathrm{C}^m(\{x\}) + c_2$$

and

$$\mathrm{C}^{c_1(m+|x|)}(\{x\}) \le \mathrm{C}^m(x) + c_2$$

for all m and x.

Indeed, having the program that decides $\{x\}$, we may try all strings in the order of increasing length until we find one that is accepted; having a program that produces x, we can compare input y with x and therefore decide $\{x\}$. In both cases we need additional space of size $O(|x|)$, so the space bound in the left-hand side is $c_1(m+x)$. In general, considering inequalities for bounded-space complexity, we use polynomially large space bounds in the left-hand side of the inequality.

Let us note, as a digression, that for time-bounded complexity such an argument does not work, and another notion of *distinguishing complexity* arises (for the complexity of a singleton $\{x\}$).

3 Space-bounded Algorithmic Statistics

Which elements of a finite set A should be considered as typical in the space-bounded setting? Let us consider the following space-bounded versions of randomness and optimality deficiencies:

$$d^a(x|A) := \log|A| - \mathrm{C}^a(x|A),$$

$$\delta^{b,d}(x, A) := \mathrm{C}^b(A) + \log|A| - \mathrm{C}^d(x).$$

Note that the optimality deficiency now uses two space bounds b and d (for A and x respectively) and is decreasing in b and increasing in d. As before, it is easy to show that both deficiencies values are non-negative (with logarithmic accuracy) provided $a \geq p(|x|)$ and $d \geq p(|x| + b)$ for a large enough polynomial p. (The term $|x|$ is needed because we enumerate elements of A in the length-lexicographic order and need to keep the last considered element.)

We may say that a set A is a good explanation for a string x (that belongs to A) if $\mathrm{C}^r(A) \approx 0$ and $\log|A| \approx \mathrm{C}^m(x)$ (with $O(\log|x|)$ accuracy) for some reasonable (not very large) values of r and m. In this case the values $d^m(x|A)$ and $\delta^{r,m}(x, A)$ are small.

As before, we can try to use a more general notion of a model and replace sets with distributions. For that we need to define the complexity of a probability distribution P with space bound m that is denoted by $\mathrm{C}^m(P)$. This value is defined as the minimal length of a program p without input and with the following two properties:

- for every x the probability of the event “p outputs x” is equal to $P(x)$;
- the memory used by p never exceeds m (however, p may use more than m random bits in general).

If such a program does not exist, then $\mathrm{C}^m(P) := \infty$.[5]

This allows us to define the optimality deficiency as

$$\delta^{b,d}(x, P) := \mathrm{C}^b(P) - \log P(x) - \mathrm{C}^d(x).$$

However, for randomness deficiency things are more complicated since it is not clear how to define conditional complexity $\mathrm{C}^m(x|P)$ if the condition P is a distribution. Moreover, even for optimality deficiency it is not obvious that it is non-negative with reasonable precision for reasonable space bounds. This can be derived from the following result (Theorem 4, but at the same time this result shows that our generalization does not give us much, as it was the case for classical algorithmic statistics (Proposition 1), though for much more subtle reasons).

[5]This is only one possible way to define the notion of bounded-space complexity for a distribution. Instead of a randomized program that has P as output distribution, we may consider a program that computes $P(x)$ for a give input x. The relations between these two definitions are not well understood.

Theorem 4 *There exist a polynomial r and a constant c such that for every string x, for every distribution P and for every m there exists a set $A \ni x$ such that $\mathrm{C}^{r(m+n)}(A) \leq \mathrm{C}^m(P) + c\log(n+m)$ and $\frac{1}{|A|} \geq P(x)2^{-c\log n}$. Here n is the length of x.*

This theorem will be proven in Section 6. The main tool is the following fact: for every constructible function $f(n) \geq \log n$ every $f(n)$-tape bounded probabilistic Turing machine can be simulated on deterministic one within $\mathrm{poly}(f(n))$ space. This result due to Simon and Jung [5].

4 Descriptions of Restricted Type

There is a problem with our version of space-bounded algorithmic statistics. It turns out that every string has a good explanation. Indeed, let x be a string such that $\mathrm{C}^m(x) = k$. Define a set $A \ni x$ as $\{y \mid \mathrm{C}^m(y) \leq k\}$. The log-size of this set is equal to k up to a constant: $\log|A| = \mathrm{C}^m(x) + O(1)$. Note also that A can be decided by a program of length $O(\log(k+m))$ that uses $\mathrm{poly}(k+m)$ space. This program includes information about m and k and tries all programs of size k using space m. (We need space k to keep track of the last program tried.)

So, for space-bounded algorithms all strings have good explanations (in other words, they are stochastic). This fact has a counterpart in classical algorithmic statistics: if we replace the complexity of A in the definition of the deficiencies by the *enumeration complexity* of A (i.e., the length of a shortest program that enumerates A), then every string x has a model of small deficiency. This model includes all strings that have complexity at most $\mathrm{C}(x)$.

So far we considered arbitrary finite sets (or more general distributions) as models (statistical hypotheses). We have seen that for such class of hypotheses the theory becomes trivial. However, in practice we usually have some a priori information about the data. We know that the data was obtained by sampling with respect to an unknown probability distribution from a known family of distributions. For simplicity we will consider only uniform distributions i.e. a family of finite sets $\mathcal{A}$.

For example, we can consider the family of all Hamming balls as $\mathcal{A}$. (That means we know a priory that our string was obtained by flipping certain number of bits in an unknown string.) Or we may consider the family that consists of all 'cylinders': for every n and for every string u of length at most n we consider the set of all n-bit strings that have prefix u. It turns out that for the first family there exists a string that has no good explanations in this family: the concatenation of the all zero string and an incompressible string (i.e. a string whose Kolmogorov complexity is close to its length) of the same length. See Proposition 13 in the appendix for a rigorous formulation and a proof.

Restricting the class of allowed hypotheses was initiated in [20]. It turns out that there exists a direct connection between randomness and optimality deficiencies in the case when a family is enumerable.

Theorem 5 ([20]) *Let $\mathcal{A}$ be an enumerable family of sets. Assume that every set from $\mathcal{A}$ consists of strings of the same length. Let x be a string of length n contained in $A \in \mathcal{A}$. Then:*

(a) $d(x|A) \le \delta(x, A) + O(\log(\mathrm{C}(A) + n))$.
(b) *There exists $B \in \mathcal{A}$ containing x such that:*
$\delta(x, B) \le d(x|A) + O(\log(\mathrm{C}(A) + n))$.

In our paper we will consider families with the following properties:

- Every set from $\mathcal{A}$ consists of strings of the same length. The family of all subsets of $\{0, 1\}^n$ that belong to $\mathcal{A}$ is denoted by $\mathcal{A}_n$.
- There exists a polynomial p such that $|\mathcal{A}_n| \le 2^{p(n)}$ for every n.
- There exists an algorithm enumerating all sets from $\mathcal{A}_n$ in space poly(n).

The last requirement means the following. There exists an indexing of $\mathcal{A}_n$ and a Turing machine M that for a pair of integers $(n; i)$ and a string x in the input, outputs 1 if x belongs to i-th set of $\mathcal{A}_n$ and 0 otherwise. Moreover, on every such input, M uses at most poly(n) space.

Any family of finite sets of strings that satisfies these three conditions is called *acceptable*. For example, the family of all Hamming balls is acceptable. Our main result is the following analogue of Theorem 5.

Theorem 6 (a) *There exist a polynomial p and a constant c such that for every set $A \ni x$ and for every m the following inequality holds*

$$d^m(x|A) \le \delta^{m,p}(x, A) + c \log(\mathrm{C}^m(A)).$$

Here $p = p(m + n)$ and n is the length of x.

(b) *For every acceptable family of sets $\mathcal{A}$ there exists a polynomial p such that the following property holds. For every $A \in \mathcal{A}$, for every $x \in A$ and for every integer m there exists a set $B \ni x$ from $\mathcal{A}$ such that*

- $\log |B| \le \log |A| + 1$;
- $\mathrm{C}^s(B) \le \mathrm{C}^m(A) - \mathrm{C}^s(A|x) + O(\log(n + m))$.

Here $s = p(m + n)$ and n is the length of x.

A critical reader may say that an analogue of Theorem 5 (b) should have the following form (and we completely agree).

Hypothesis 1 *There exist a polynomial p and a constant c such that for every set $A \ni x$ from $\mathcal{A}$ and for every m there exists a set $B \in \mathcal{A}$ such that*

$$\delta^{p,m}(x, B) \le d^p(x|A) + c \log(n + m).$$

Here $p = p(m + n)$, n is the length of x and $\mathcal{A}$ is an acceptable family of sets.

We argue in Section 5.1 why Theorem 6 (b) is close to Hypothesis 1.

5 Proof of Theorem 6

Proof of Theorem 6(a) The inequality we have to prove is equivalent to the following

$$\mathrm{C}^p(x) \leq \mathrm{C}^m(x|A) + \mathrm{C}^m(A) + c\log(\mathrm{C}^m(A) + n)$$

(by the definitions of optimality and randomness deficiencies).

Consider a program p of length $\mathrm{C}^m(x|A)$ that distinguishes x and uses A as an oracle. We need to construct a program that also distinguishes x but does not use any oracle. For this add to p a procedure distinguishing A. There exists such a procedure of length $\mathrm{C}^m(A)$. So, we get a program of the required length (with an additional $O(\log(\mathrm{C}^m(A)))$ bits used for pair coding) that uses poly(m) space. □

For all x and $A \ni x$ the randomness deficiency is not greater than the optimal deficiency. The following example shows that the difference can be large.

Example 1 Consider an incompressible string x of length n, thus $\mathrm{C}(x) = n$ (this equality as well as further ones holds with logarithmic precision). Let y be an n-bit string that is also incompressible and independent of x, i.e. $\mathrm{C}(y|x) = n$. By symmetry of information (see [7, 16]) we have $\mathrm{C}(x|y) = n$.

Define $A := \{0,1\}^n \setminus \{y\}$. The randomness deficiency of x in A (without resource restrictions) is equal to 0. Hence, this is true for any resource restrictions ($\mathrm{C}(x|A)$ is not greater than $\mathrm{C}^m(x|A)$ for every m). Hence, for any m we have $d^m(x|A) = 0$. On the other hand $\delta^{p,m}(x, A) = n$ for all p and large enough m. Indeed, take $m =$ poly(n) such that $\mathrm{C}^m(x) = n$. Since $\mathrm{C}(A) = n$ we have $\mathrm{C}^q(A) = n$ for every q.

So, we can not just let $A = B$ in Hypothesis 1. In some cases we have to 'improve' A (in the example above we can take $\{0,1\}^n$ as an improved set).

5.1 Sketch of the Proof of Theorem 5(b)

The proof of Theorem 6 (b) is similar to the proof of Theorem 5 (b). We present a sketch of the proof of Theorem 5 (b).

Theorem 5 (b) states that there exists a set $B \in \mathcal{A}$ containing x such that $\delta(x, B) \leq d(x|A)$. (Here and later we omit terms of logarithmic order.) We derive this from the following statement.

(1) There exists a set $B \in \mathcal{A}$ containing x such that

$$|B| \leq 2 \cdot |A| \text{ and } \mathrm{C}(B) \leq \mathrm{C}(A) - \mathrm{C}(A|x).$$

For such B the inequality $\delta(x, B) \leq d(x|A)$ easily follows from the inequality $\mathrm{C}(A) - \mathrm{C}(A|x) - \mathrm{C}(x) \leq -\mathrm{C}(x|A)$. The latter inequality holds by symmetry of information.

To prove (1) note that

(2) there exist at least $2^{\mathrm{C}(A|x)}$ sets in $\mathcal{A}$ containing x whose complexity and size are at most $\mathrm{C}(A)$ and $2 \cdot |A|$, respectively.

Indeed, knowing x we can enumerate all sets from $\mathcal{A}$ containing x whose parameters (complexity and size) are not worse than the parameters of A. Since we can describe A by its ordinal number in this enumeration we conclude that the length of this number is at least $\mathrm{C}(A|x)$ (with logarithmic precision).

Now (1) follows from the following statement.

(3) Assume that $\mathcal{A}$ contains at least 2^k sets of complexity at most i and size at most 2^j containing x. Then one of them (i.e., some set in $\mathcal{A}$ of size at most 2^j containing x) has complexity at most $i-k$.

(We will apply it to $i = \mathrm{C}(A)$, $j = \lceil \log |A| \rceil$ and $k = \mathrm{C}(A|x)$.)

So, Theorem 6 (b) is an analogue of (2). Despite there is an analogue of symmetry of information for space-bounded algorithms (see [8] and Appendix) Hypothesis 1 does not follow from Theorem 6 (b) directly. The problem is the following. For inequalities for space-bounded complexity we can not sum inequalities and derive from them new inequalities as in the unbounded case. Consider the following example. Let X, Y, Z, W and T be some numbers. Assume that $X + Y \le Z$ and $Z + W \le Y + T$. Then of course $X + W \le T$. Now let X, Y, Z, W and T be not numbers but functions depending on some space bounds. In our case $X = \mathrm{C}(B)$, $Y = \mathrm{C}(A|x)$, $Z = \mathrm{C}(A)$, $W = \mathrm{C}(x|A)$ and $T = \mathrm{C}(x)$. The inequality above holds in the following form:

$$\forall p \exists q : \ Z^q + W^q \le Y^p + T^p \text{ and } \forall q \exists r : \ X^r + Y^r \le Z^q.$$

Then we can derive that

$$\forall p \exists r : \ X^r + Y^r + W^q \le Y^p + T^p.$$

This is not what we want because Y^r can be smaller than Y^p.

In the next subsection we derive Theorem 6 (b) from Lemma 7 (this is an analogue of the third statement). In the proof of Lemma 7 we use the Nisan-Wigderson generator.

5.2 Main Lemma

We will derive Theorem 6 (b) from the following

Lemma 7 *Fix some acceptable family $\mathcal{A}$ of sets. Then there exist a polynomial p and a constant c such that the following statement holds for every j:*

For every string x of length n that belongs to at least 2^k sets from $\mathcal{A}_n$ that have cardinality at most 2^j and space-bounded complexity (with bound m) at most i, there is a set $A \in \mathcal{A}_n$ that contains x, has cardinality at most 2^j and space-bounded complexity at most $i - k + c\log(n+m)$ with bound $m + p(n)$.

Proof of Theorem 6 (b) assuming Lemma 7 Denote by $\mathcal{A}'$ the family of all sets in $\mathcal{A}_n$ containing x whose parameters are not worse than those of A.

$$\mathcal{A}' := \{A' \in \mathcal{A}_n \mid x \in A', \mathrm{C}^m(A') \le \mathrm{C}^m(A), \log|A'| \le \lfloor \log |A| \rfloor\}.$$

Let $k = \log |\mathcal{A}'|$.

We describe A with $k + O(\log(n+m))$ bits when x is known. The sets in $\mathcal{A}'$ (more specifically, their programs) can be enumerated given n, m and $\lfloor \log |A| \rfloor$ are known.

This enumeration can be done in space poly$(m+n)$. We describe A by its ordinal number in this enumeration, thus

$$\mathrm{C}^s(A|x) \leq k + O(\log(n+m)).$$

Here $s = \mathrm{poly}(m+n)$.

Theorem 6 (b) follows from Lemma 7 for $i = \mathrm{C}^m(A)$ and $j = \lfloor \log |A| \rfloor$. □

5.3 Nisan-Wigderson Generator. Proof of the main Lemma

Define

$$\mathcal{A}_{n,m}^{i,j} := \{A' \in \mathcal{A}_n \mid \mathrm{C}^m(A') \leq i, \log |A'| \leq j\}$$

for an acceptable family of sets $\mathcal{A}$. Define a random family of sets $\mathcal{B}$ as follows: every set from $\mathcal{A}_{n,m}^{i,j}$ belongs to $\mathcal{B}$ with probability $2^{-k}(n+2)\ln 2$ independently. We claim that $\mathcal{B}$ satisfies the following two properties with high probability.

(1) The cardinality of $\mathcal{B}$ is at most $2^{i-k+2} \cdot (n+k)^2 \ln 2$.
(2) If a string of length n is contained in at least 2^k sets from $\mathcal{A}_{n,m}^{i,j}$ then one of these sets belongs to $\mathcal{B}$.

Lemma 8 *With probability at least $\frac{1}{2}$, family $\mathcal{B}$ satisfies both properties* (1) *and* (2).

Proof We show that $\mathcal{B}$ satisfies each of these properties (separately) with probability at least $\frac{3}{4}$.

For property (1), this follows from Markov's inequality: the cardinality of $\mathcal{B}$ exceeds the expectation by a factor of 4 with probability less than $\frac{1}{4}$. (We can get a much smaller bound.)

For property (2), consider a string of length n that belongs to at least 2^k sets from $\mathcal{A}_{n,m}^{i,j}$. The probability of the event that all of these 2^k sets do not belong to $\mathcal{B}$, is at most

$$(1 - 2^{-k}(n+2)\ln 2)^{2^k} \leq 2^{-n-2} \text{ (since } 1 - x \leq e^{-x}).$$

The probability of the sum of such events for all strings of length n is at most $2^n 2^{-n-2} = \frac{1}{4}$. □

Using Lemma 8 we prove the existence of a set whose *unbounded* complexity is at most $i - k + O(\log(n+m))$. Indeed, by Lemma 8 *there exists* a subfamily that satisfies properties (1) and (2). The lexicographically first such family has small complexity, because it can be computed given i, k, n and m. Note, that k and i are bounded by poly(n): since $\mathcal{A}$ is acceptable, $\log |\mathcal{A}_n| = \mathrm{poly}(n)$, and hence, k is at most poly(n). We can enumerate all sets from $\mathcal{A}_n$, thus the space-bounded complexity of every element of $\mathcal{A}_n$ (and in particular, i) is bounded by a polynomial in n. Now we can describe the required set by the ordinal number of an enumeration of this subfamily.

However, this method does not satisfy the polynomial space-bounded in the complexity terms: the brute-force search to find a suitable subfamily requires exponential space. To reduce this space, we use the Nisan-Wigderson generator. A similar idea was used in [11].

Theorem 9 ([12, 13]) *For every constant d and every positive polynomial $q(m)$ there exists a sequence of functions $G_m : \{0,1\}^f \to \{0,1\}^m$ where $f = O(\log^{2d+6} m)$ such that:*

- *Function G_m is computable in space poly(f);*
- *For every family of circuits C_m of size $q(m)$ and depth d and for large enough m it holds that:*

$$|\Pr_x[C_m(G_m(x)) = 1] - \Pr_y[C_m(y) = 1]| < \frac{1}{m},$$

where x is distributed uniformly in $\{0,1\}^f$, and y is distributed uniformly in $\{0,1\}^m$.

We will use this theorem with $m = 2^{i+n}$. Then f is a polynomial in $i + n$ (for constant d), hence $f = \text{poly}(n)$. Every element whose complexity is at most i corresponds to a string of length i in the natural way. So, we can assign subfamilies of $\mathcal{A}^{i,j}_{n,m}$ to strings of length m.

Assume that there exists a circuit of size $2^{O(n)}$ and constant depth that inputs a subfamily of $\mathcal{A}^{i,j}_n$ and outputs **1** if this subfamily satisfies properties (1) and (2) from Lemma 8, and **0** otherwise. First we prove Lemma 7 using this assumption.

Compute $G_m(y)$ for all strings y of length f until we find a suitable one, i.e., whose image satisfies the two properties. Such a string exists by Lemma 8, Theorem 9 and our assumption. Note that we can find the lexicographically first suitable string by using space $m + \text{poly}(n)$, so *bounded by space* $m + \text{poly}(n)$ the complexity of this string is equal to $O(\log(n + m))$.

If we construct a constant depth circuit of the required size that verifies properties (1) and (2), we are finished. Unfortunately we do not know how to construct such a circuit verifying the first property (we can not compute threshold functions by constant-depth circuits—see [4]). However, we use the following:

Theorem 10 ([1]) *For every t there exists a circuit of constant depth and poly(t) size that on input a binary string of length t outputs 1 if the input has at most $\log^2 t$ ones and outputs 0 otherwise.*

To use this theorem we make a little change of the first property. Divide $\mathcal{A}^{i,j}_n$ into 2^{i-k} parts of size 2^k (according to an algorithm that enumerates $\mathcal{A}^{i,j}_n$). The corrected property is the following.

(1)* The family of sets $\mathcal{B}$ contains at most $(n + k)^2$ sets from each of these parts.

Lemma 11 *The family of sets $\mathcal{B}$ satisfies both properties* (1)* *and* (2) *with probability at least $\frac{1}{3}$.*

The proof of this lemma is not difficult but uses some cumbersome formulas and is given in the Appendix.

Proof of Lemma 7 It is clear that property (1)* implies property (1). By using Lemma 7 and the discussion above, it is enough to show that properties (1)* and (2) can be verified by constant depth circuits of size $2^{O(i+n)}$.

Such a circuit exists for property (1)* by Theorem 10.

The second property can be verified by the following circuit of depth 2. For every string of length n contained in 2^k sets from $\mathcal{A}_n^{i,j}$, construct a corresponding disjunct. Then construct a conjunction gate uniting all of these disjuncts. □

6 Proof of Theorem 4

Theorem 4 has an easy proof under the assumption that the program corresponding to distribution P uses poly(n) random bits. Indeed, in such case we can run a program with all possible random bits and calculate $P(x)$ in polynomial space. Hence, we can describe A as the set of all strings for which $2^{-k} \geq P(x) > 2^{-k-1}$.

The proof of this theorem for the general case (when the number of random bits is exponentially large) was sketched in [3] (answering the author's question). We provide a more accessible argument using the following theorem.

Theorem 12 ([5]) *Let f be a probabilistic program that uses at most $r(n)$ space on inputs of length n for some polynomial r. Then there exists a deterministic program $\widehat{f}$ with the following properties:*

(a) *$\widehat{f}$ uses at most $r^2(n)$ space on inputs of length n;*
(b) *if $Pr[f(x) = 1] > \frac{2}{3}$ then $\widehat{f}(x) = 1$. If $Pr[f(x) = 1] < \frac{1}{3}$ then $\widehat{f}(x) = 0$;*
(c) *$|\widehat{f}| \leq |f| + O(1)$.* [6]

Proof of Theorem 4 If the complexity of distribution P (bounded by space m) is equal to infinity, then $A = \{x\}$ satisfies the conditions of the theorem.

Otherwise, P can be specified by a program g. Consider the integer k such that: $2^{-k+1} \geq P(x) \geq 2^{-k}$. We can assume that k is not greater than n—the length of x—, otherwise, $A = \{0, 1\}^n$ satisfies the conditions of the theorem.

Note, that we can find a good approximation for $P(y)$ by running g exponentially many times.

More accurately, we evaluate g, 2^{100k^2} times. For every string y, let $\omega(y)$ be the frequency of the output of y. The following inequality holds by Hoeffding's inequality

$$\Pr[|\omega(y) - P(y)| > 2^{-k-10}] < \frac{1}{3}.$$

Hence, using program g, we can construct a program f that uses poly(n) space (on inputs of length n) such that

[6] We use a stronger variant than the theorem in [5], but the proof is the same: we added requirement (c), but this is easily seen to be true, because the constructed program for $\hat{f}$ is a simple transformation of f, and it suffices to add some fixed amount of instructions to f. Also, the theorem in [5] does not assume that Pr$[f(x)]$ belongs to $[\frac{1}{3}; \frac{2}{3}]$. However, this assumption is not used in the proof of this theorem.

(1) if $P(y) > 2^{-k-1}$ and $|y| = n$ then $\Pr[f(y) = 1] > \frac{2}{3}$;
(2) if $P(y) < 2^{-k-2}$ then $\Pr[f(y) = 0] > \frac{2}{3}$.

Using Theorem 12 for f we get a program $\widehat{f}$ such that $|\widehat{f}| \leq |g| + O(\log n)$. By the first property of f we get $\widehat{f}(x) = 1$. From the second property it follows that the cardinality of the set $\{y \mid \widehat{f}(y) = 1\}$ is not greater than 2^{k+2}. Thus, this set satisfies the requirements of the theorem. □

Open Question

Does Hypothesis 1 hold?

Acknowledgments I would like to thank Bruno Bauwens, Ricky Demer, Nikolay Vereshchagin and Alexander Shen for useful discussions, advice and remarks.

This work is supported in parts by the RFBR grant 16-01-00362, by the Young Russian Mathematics award, MK-5379.2018.1 and the RaCAF ANR-15-CE40-0016-01 grant. The study has also been funded by the Russian Academic Excellence Project '5-100'.

Appendix

Proposition 13 *Let $x = y \cdot z$ be a concatenation of strings y and z of length n where $y = \overbrace{000\ldots00}^{n\ zeros}$ and $\mathrm{C}(z) > n - \varepsilon$ for some positive ε. Assume that x belongs to some Hamming ball B. Then*

$$2\mathrm{C}(B) + \log|B| - \mathrm{C}(x) > \frac{2}{5}n - \varepsilon - O(\log n).$$

If ε is small, then Hamming ball B can not satisfy $\mathrm{C}(B) \approx 0$ and $\log|B| \approx \mathrm{C}(x)$.

Proof Denote by r and b the radius and the center of B. Denote by b_0 and b_1 the first and the second parts of b. (So, $b = b_0 \cdot b_1$ and $|b_0| = |b_1| = n$.) Then z belongs to the Hamming ball B_1 with center b_1 and radius r. Hence,

$$\mathrm{C}(B_1) + \log|B_1| \geq \mathrm{C}(z) - O(\log n) = \mathrm{C}(x) - O(\log n).$$

Note that $\mathrm{C}(B_1) \leq \mathrm{C}(B) + O(\log n)$. The log-size of B_1 can be estimated as $nH(\frac{r}{n}) + O(\log n)$, where $H(t) := -t\log t - (1-t)\log(1-t)$ is the binary Shannon entropy (see, for example, [9]). So, the inequality above can be rewritten as

$$\mathrm{C}(B) + nH\left(\frac{r}{n}\right) \geq \mathrm{C}(x) - O(\log n). \tag{1}$$

We need to show that $2\mathrm{C}(B) + \log|B| - \mathrm{C}(x) > \frac{2}{5}n - \varepsilon - O(\log n)$, where $\log|B| = 2nH(\frac{r}{2n}) + O(\log n)$. This inequality follows from (1) and the inequality below by an easy calculation.

$$H\left(\frac{r}{n}\right) - H\left(\frac{r}{2n}\right) \leq \frac{3}{10}.$$

To verify the last inequality one can show that the maximum of the function $H(t) - H(\frac{t}{2})$ is equal to $\frac{\ln(\frac{3}{4}+\frac{1}{\sqrt{2}})}{\ln 4} < \frac{3}{10}$. □

Symmetry of Information

Define $\mathrm{C}^m(A, B)$ as the minimal length of a program that on input a pair of strings (a, b) uses at most space m, and outputs the bits $(a \in A, b \in B)$.

Lemma 14 (Symmetry of information) *Assume $A, B \subseteq \{0, 1\}^n$. Then*

$$\text{(a) } \forall m\ \mathrm{C}^p(A, B) \le \mathrm{C}^m(A) + \mathrm{C}^m(B|A) + O(\log(\mathrm{C}^m(A, B) + m + n))$$

for $p = m + poly(n + \mathrm{C}^m(A, B))$.

$$\text{(b) } \forall m\ \mathrm{C}^p(A) + \mathrm{C}^p(B|A) \le \mathrm{C}^m(A, B) + O(\log(\mathrm{C}^m(A, B) + m + n))$$

for $p = 2m + poly(n + \mathrm{C}^m(A, B))$.

Proof of Lemma 14 (a) The proof is similar to the proof of Theorem 6 (a). □

Proof of Lemma 14 (b) Let $k := \mathrm{C}^m(A, B)$. Denote by $\mathcal{D}$ the family of sets (U, V) such that $\mathrm{C}^m(U, V) \le k$ and $U, V \subseteq \{0, 1\}^n$. It is clear that $|\mathcal{D}| < 2^{k+1}$. Denote by $\mathcal{D}_A$ the pairs of $\mathcal{D}$ for which the first element is equal to A. Let t satisfy the inequalities $2^t \le |\mathcal{D}_A| < 2^{t+1}$.

We prove that

- $\mathrm{C}^p(B|A)$ does not exceed t significantly;
- $\mathrm{C}^p(A)$ does not exceed $k - t$ significantly.

Here $p = 2m + O(n)$.

We start with the first statement. There exists a program that enumerates all sets from $\mathcal{D}_A$ using A as an oracle and that works in space $2m + O(n)$. Indeed, such enumeration can be done in the following way: enumerate all programs of length k and verify the following condition for every pair of n-bit strings. First, a program uses at most m space on this input and does not loop. To verify it we need to check that the program does not compute longer than $2^{O(m)}$ steps. Second, if a second n-bit string belongs to A then the program outputs 1, and 0 otherwise. Append to this program the ordinal number of a program that distinguishes (A, B). This number is not greater than $t+1$. Therefore we have $\mathrm{C}^p(B|A) \le t + O(\log(\mathrm{C}^m(A, B)+m+n))$.

Now we prove the second statement. Note that there exist at most 2^{k-t+1} sets U such that $|\mathcal{D}_U| \ge 2^t$ (including A). Hence, if we construct a program that enumerates all sets with such property (and does not use much space) then we are finished, because the set A can be described by the ordinal number of this enumeration. Let us construct such a program. It works as follows:

Enumerate all sets U that are the first elements from $\mathcal{D}$, i.e. we enumerate programs that distinguish the corresponding sets (say, lexicographically). We go to the next step if the following properties hold. First, $|\mathcal{D}_U| \ge 2^t$, and second: we did not consider set U earlier (i.e. every program whose lexicographical number is smaller does not distinguish U or is not the first element from a set from $\mathcal{D}$).

This program uses $2m+\text{poly}(n+\mathrm{C}^m(A, B))$ space and has length $O(\log(\mathrm{C}^m(A)+n+m))$, and hence satisfies all requirements. $\square$

Proof of Lemma 11 It suffices to show that $\mathcal{B}$ satisfies property (1)* with probability at most 2^{-n}, because $\mathcal{B}$ satisfies property (2) with probability at most $\frac{1}{4}$.

For this let us show that every part is 'bad' (i.e. has at least $(n+k)^2+1$ sets from $\mathcal{B}$) with probability at most 2^{-2n}. The probability of such event is equal to the probability that a binomial random variable with parameters $(2^k, 2^{-k}(n+2)\ln 2)$ exceeds $(n+k)^2$. To bound this, we use an easy but lengthy sequence of estimations. For $w := 2^k$, $p := 2^{-k}(n+2)\ln 2$ and $v := (n+k)^2$ we have

$$\sum_{i=v}^{w}\binom{w}{i}p^i(1-p)^{w-i} < w\cdot\binom{w}{v}p^v(1-p)^{w-v} < w\cdot\binom{w}{v}p^v < w\frac{(wp)^v}{v!}.$$

The leftmost inequality follows from $wp = (n+2)\ln 2 \le (n+k)^2 = v$. Because $wp = (n+2)\ln 2 < 10n$, we obtain

$$w\frac{(wp)^v}{v!} < \frac{2^k(10n)^{(n+k)^2}}{((n+k)^2)!} \ll 2^{-2n}.$$

$\square$

References

1. Ajtai, M.: Approximate counting with uniform constant-depth circuits, advances in computational complexity theory. In: DIMACS series in discrete mathematics and theoretical computer science, american mathematical society, pp. 1–20 (1993)
2. Buhrman, H., Fortnow, L., Laplante, S.: Resource-bounded Kolmogorov complexity revisited. SIAM J. Comput. **31**(3), 887–905 (2002)
3. Demer, R.: Stack exchange discussion (Ricky Demer's answer to the author's question) http://cstheory.stackexchange.com/questions/34896/can-every-distribution-producible-by-a-probabilistic-pspace-machine-be-produced (2016)
4. Furst, M., Saxe, J.B., Sipser, M.: Parity, circuits, and the polynomial-time hierarchy. Mathematical Systems Theory **17**(1), 13–27 (1984)
5. Jung, H.: Relationships between probabilistic and deterministic tape complexity. In: Mathematical foundations of computer science 1981 (MFCS 1981), Lecture Notes in Computer Science, vol. 118, pp. 339–346 (1981)
6. Kolmogorov, A.N.: Three approaches to the quantitative definition of information. Probl. Inf. Transm. **1**(1), 4–11 (1965). English translation published in: International Journal of Computer Mathematics, 2, pp 157–168 (1968)
7. Li, M., Vitányi, P.M.B.: An introduction to Kolmogorov complexity and its applications, 3rd edn. Springer, Berlin (2008)
8. Longpré, L.: Resource bounded Kolmogorov complexity, a link between computational complexity and information theory, Ph.D. thesis, TR86-776. Cornell University, Ithaca (1986)
9. MacWilliams, F.J., Sloane, N.J.A.: The theory of error-correcting codes. I and II. Bull. Amer. Math. Soc. **84**(6), 1356–1359 (1978)
10. Milovanov, A.: On algorithmic statistics for space-bounded algorithms. In: Proceedings of 12th international computer science symposium in Russia (CSR 2017) LNCS, vol. 10304, pp. 232–234 (2017)
11. Musatov, D.: Improving the space-bounded version of Muchnik's conditional complexity theorem via "naive" derandomization. Theory of computing systems **55**(2), 299–312 (2014)
12. Nisan, N.: Pseudorandom bits for constant depth circuits. Combinatorica **11**, 63–70 (1991)

13. Nisan, N., Wigderson, A.: Hardness vs randomness. J. Comput. Syst. Sci. **49**(2), 149–167 (1994)
14. Shen, A.: Around Kolmogorov complexity: basic notions and results. In: Vovk, V., Papadoupoulos, H., Gammerman, A. (eds.) Measures of Complexity. Festschrift for Alexey Chervonenkis. Springer, Berlin (2015). ISBN: 978-3-319-21851-9
15. Shen, A.: The concept of (α, β)-stochasticity in the Kolmogorov sense, and its properties. Soviet Mathematics Doklady **271**(1), 295–299 (1983)
16. Shen, A., Uspensky V., Vereshchagin N.: Kolmogorov complexity and algorithmic randomness, MCCME. (Russian). English translation: http://www.lirmm.fr/~ashen/kolmbook-eng.pdf (2013)
17. Sipser, M.: A complexity theoretic approach to randomness. In: Proceedings of the 15th ACM symposium on the theory of computing, pp. 330–335 (1983)
18. Vereshchagin, N., Shen, A.: Algorithmic statistics: forty Years. In: Computability and complexity. Essays Dedicated to Rodney G. Downey on the Occasion of His 60Th Birthday. LNCS, vol. 10010, pp. 669–737. Springer, Heidelberg (2017)
19. Vereshchagin, N., Vitányi, P.M.B.: Kolmogorov's structure functions with an application to the foundations of model selection. IEEE Trans. Inf. Theory **50**(12), 3265–3290 (2004). Preliminary version: Proceedings of 47th IEEE Symposium on the Foundations of Computer Science, pp. 751–760 (2002)
20. Vereshchagin, N., Vitányi, P.M.B.: Rate distortion and denoising of individual data using Kolmogorov complexity. IEEE Trans. Inf. Theory **56**(7), 3438–3454 (2010)