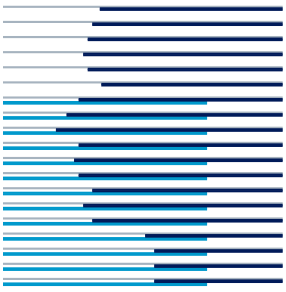
On computability of pattern recognition problems

Daniil Ryabko
daniil@ryabko.net

Technical Report No. IDSIA-17-05
July 19, 2005



IDSIA / USI-SUPSI
Dalle Molle Institute for Artificial Intelligence
Galleria 2, 6928 Manno, Switzerland

On computability of pattern recognition problems

Daniil Ryabko *
daniil@ryabko.net

July 19, 2005

Abstract

In statistical setting of the pattern recognition problem the number of examples required to approximate an unknown labelling function is linear in the VC dimension of the target learning class. In this work we consider the question whether such bounds exist if consider only computable pattern recognition methods, assuming that the unknown labelling function is also computable. We find that in this case the number of examples required for a computable method to approximate the labelling function not only is not linear, but grows faster (in the VC dimension of the class) than any computable function. No time or space constraints are put on the predictors or target functions; the only resource we consider is the training examples.

The task of pattern recognition is considered in conjunction with another learning problem — data compression. An impossibility result for the task of data compression allows us to estimate the sample complexity for pattern recognition.

1 Introduction

The task of pattern recognition consists in predicting an unknown label of some observation (or object). For instance, the object can be an image of a hand-written letter, in which case the label is the actual letter represented by

*This work was supported by SNF grant 200020-107616

this image. Other examples include DNA sequence identification, recognition of an illness based on a set of symptoms, speech recognition, and many others.

More formally, the objects are drawn independently from the object space X (usually $X = [0,1]^d$ or $\mathbb{R}^d$) according to some unknown but fixed probability distribution P on X, and labels are defined according to some function $\eta : X \to Y$, where Y is a finite set (often $Y = \{0,1\}$). The task is to construct a function $\varphi : \{0,1\}^* \to Y$ which approximates η, i.e. for which $P\{x : \eta(x) \neq \varphi(x)\}$ is small, where P and η are unknown but examples $x_1, y_1, \ldots, x_n, y_n$ are given; $y_i := \eta(x_i)$. In the framework of statistical learning theory [7],[8] it is assumed that the function η belongs to some known class of functions $\mathcal{C}$. Good error estimated can be obtained if the class $\mathcal{C}$ is small enough. More formally, the number of examples required to obtain a certain level of accuracy (or the *sample complexity* of $\mathcal{C}$) is linear in the VC-dimension of $\mathcal{C}$.

In this work we investigate the question whether such bounds can be obtained if we consider only computable (on some Turing machine) pattern recognition methods. To make the problem more realistic, we also assume that the target function η is also computable. Both the predictors and the target functions are of the form $\{0,1\}^\infty \to \{0,1\}$.

We show that there are classes $\mathcal{C}_k$ of functions for which the number of examples needed to approximate the pattern recognition problem to a certain accuracy grows faster in the VC dimension of the class than any computable function (rather than being linear as in the statistical setting). In particular this holds if $\mathcal{C}_k$ is the class of all computable functions of length not greater than k.

Importantly, the same negative result holds even if we allow the data to be generated "actively", e.g. by some algorithm, rather than just by some fixed probability distribution.

To obtain this negative result we consider the task of data compression: an impossibility result for the task of data compression allows us to estimate the sample complexity for pattern recognition. We also analyze how tight is the negative result, and show that for some simple computable rule (based on the nearest neighbour estimate) the sample complexity is finite in k, under different definitions of computational patterning recognition task.

In comparison to the vast literature on pattern recognition and related learning problems relatively little attention had been paid to the "computable" version of the task; at least this concerns the task of approximating any computable function. There is a track of research in which different

concepts of computable learnability of functions on countable domains are studied, see [2]. A link between this framework and statistical learning theory is proposed in [5], where it is argued that for a uniform learnability finite VC dimension is required.

Another approach is to consider pattern recognition methods as functions computable in polynomial time, or under other resource constraints. This approach leads to many interesting results, but it usually considers more specified settings of a learning problem, such as learning DNFs, finite automata, etc. See [3] for an introduction to this theory and for references.

2 Preliminaries

A *(binary) string* is a member of the set $\{0,1\}^* = \cup_{i=0}^{\infty}\{0,1\}^n$. The length of a string x will be denoted by $|x|$, while x^i is the ith element of x, $1 \leq i \leq |x|$. For a set A the symbol $|A|$ is used for the number of elements in A. We will assume the lexicographical order on the set of strings, and when necessary will identify $\{0,1\}^*$ and $\mathbb{N}$ via this ordering. Let $\mathbb{N}$ be the sets of natural numbers. The symbol log is used for $\log_2$. For a real number α the symbol $\ulcorner\alpha\urcorner$ is the least natural number not smaller than α.

In pattern recognition a labelling function is usually a function from the interval $[0,1]$ or $[0,1]^d$ (sometimes more general spaces are considered) to a finite space $Y := \{0,1\}$. As we are interested in computable functions, we consider instead the functions of the form $\{0,1\}^\infty \to \{0,1\}$. Moreover, we call a partial recursive function (or program) η a *labelling function* if there exists such $t =: t(\eta) \in \mathbb{N}$ that η accepts all strings from $X_t := \{0,1\}^t$ and only such strings [1]. For an introduction to the computability theory see for example [6].

It can be argued that this definition of a labelling function is too restrictive to approximate well the notion of a real function. However, as we are after negative results (for the class of all labelling functions), it is not a disadvantage. Other possible definitions are discussed in Section 4, where we are concerned with tightness of our negative results.

All computable function can be encoded (in a canonical way) and thus

[1] It is not essential for this definition that η is not a total function. An equivalent (for our purposes) definition would be as follows. A labelling function is any total function which outputs the string 00 on all inputs except on the strings of some length $t =: t(\eta)$, on each of which it outputs either 0 or 1.

the set of computable functions can be effectively enumerated. Define the *length* of η as $l(\eta) := |n|$ where n is the number of η in such enumeration.

Define the task of computational pattern recognition as follows. An (unknown) labelling function η is fixed. The objects $x_1, \ldots, x_n \in X$ are drawn according to some distribution P on $X_{t(\eta)}$. The labels y_i are defined according to η, that is $y_i := \eta(x_i)$.

A *predictor* is a family of functions (indexed by n)

$$\varphi_n(x_1, y_1, \ldots, x_n, y_n, x),$$

taking values in Y, such that for any n and any $t \in \mathbb{N}$, if $x_i \in X_t$ for each i, $1 \leq i \leq n$, then the marginal $\varphi(x)$ is a total recursive function on X_t (that is, $\varphi_n(x)$ accepts any $x \in X_t$). We will often identify φ_n with its marginal $\varphi_n(x)$ when the values of other variables are clear.

Thus, given a *sample* $x_1, y_1, \ldots, x_n, y_n$ of labelled objects of the same size t, a predictor produces a computable function; this function is supposed to approximate the labelling function η on X_t.

A *computable predictor* is a predictor which for any $t \in \mathbb{N}$ and any $n \in \mathbb{N}$ is a total recursive function on $X_t \times Y \times \cdots \times X_t \times Y \times X_t$

3 Main results

We are interested in what size sample is required to approximate a labelling function η. Moreover, for a (computable) predictor φ, a labelling function η and $0 < \varepsilon \in \mathbb{R}$ define

$$\delta_n(\varphi, \eta, \varepsilon) := \sup_{P_t} P_t\Big\{x_1, \ldots, x_n \in X_t : \\ P_t\big\{x \in X_t : \varphi_n(x_1, y_1, \ldots, x_n, y_n, x) \neq \eta(x)\big\} > \varepsilon\Big\},$$

where $t = t(\eta)$ and P_t ranges over all distributions on X_t. For $\delta \in \mathbb{R}$, $\delta > 0$ define the *sample complexity* of η with respect to φ as

$$N(\varphi, \eta, \delta, \varepsilon) := \min\{n \in \mathbb{N} : \delta_n(\varphi, \eta, \varepsilon) \leq \delta\}.$$

The number $N(\varphi, \eta, \delta, \varepsilon)$ is the minimal sample size required for a predictor φ to achieve ε-accuracy with probability $1 - \delta$ when the (unknown) labelling function is η.

We can use statistical learning theory [7] to derive the following statement

Proposition 1. *There exists a predictor φ such that*

$$N(\varphi, \eta, \delta, \varepsilon) \leq \max\left(l(\eta)\frac{8}{\varepsilon}\log\frac{13}{\varepsilon}, \frac{4}{\varepsilon}\log\frac{2}{\delta}\right)$$

for any labelling function η and any $\varepsilon, \delta > 0$.

Observe that the bound is linear in the length of η.

In what follows the proof of this simple statement, we investigate the question of whether any such bounds exist if we restrict our attention to computable predictors.

Proof. The predictor φ is defined as follows. For each sample $x_1, y_1, \ldots, x_n, y_n$ it finds a shortest program $\bar{\eta}$ such that $\bar{\eta}(x_i) = y_i$ for all $i \leq n$. Clearly, $l(\bar{\eta}) \leq l(\eta)$. Observe that the VC-dimension of the class of all functions of length not greater than $l(\eta)$ is bounded from above by $l(\eta)$, as there are not more than $2^{l(\eta)}$ such functions. Moreover, φ minimises empirical risk over this class of functions. It remains to use the following bound (see e.g. [1], Corollary 12.4)

$$\sup_{\eta\in\mathcal{C}} N(\varphi, \eta, \delta, \varepsilon) \leq \max\left(V(\mathcal{C})\frac{8}{\varepsilon}\log\frac{13}{\varepsilon}, \frac{4}{\varepsilon}\log\frac{2}{\delta}\right)$$

where $V(\mathcal{C})$ is the VC-dimension of the class $\mathcal{C}$. □

The main result of this work is that for any computable predictor φ there is no computable upper bound in terms of $l(\eta)$ on the sample complexity of the function η with respect to φ:

Theorem 1. *For any computable predictor φ and any total recursive function $\beta : \mathbb{N} \to \mathbb{N}$ there exist a labelling function η, and some $n > \beta(l(\eta))$ such that*

$$P\{x \in X_{t(\eta)} : \varphi(x_1, y_1, \ldots, x_n, y_n, x) \neq \eta(x)\} > 0.05,$$

for any $x_1, \ldots, x_n \in X_{t(\eta)}$, where $y_i = \eta(x_i)$ and P is the uniform distribution on $X_{t(\eta)}$.

For example, we can take $\beta(n) = 2^n$, or 2^{2^n}.

Corollary 1. *For any computable predictor φ, any total recursive function $\beta : \mathbb{N} \to \mathbb{N}$ and any $\delta < 1$*

$$\sup_{\eta: l(\eta)\leq k} N(\varphi, \eta, \delta, 0.05) > \beta(k)$$

from some k on.

Observe that there is no δ in the formulation of Theorem 1. Moreover, it is not important how the objects $(x_1, \ldots, x_n)$ are generated — it can be any individual sample. In fact, we can assume that the sample is chosen in any manner, for example by some algorithm. This means that no computable upper bound on sample complexity exists even for *active learning algorithms*.

It appears that the task of pattern recognition is closely related to another learning task — data compression. Moreover, to prove Theorem 1 we need a similar negative result for this task. Thus before proceeding with the proof of the theorem, we introduce the task of data compression and derive some negative results for it. We call a total recursive function $\psi : \{0,1\}^* \to \{0,1\}^*$ an *data compressor* if it is an injection (i.e. $x_1 \neq x_2$ implies $\psi(x_1) \neq \psi(x_2)$). We say that an data compressor *compresses* the string x if $|\psi(x)| < |x|$. Clearly, for any natural n any data compressor compresses not more than a half of strings of size not greater than n.

We will now present a definition of Kolmogorov complexity; for fine details see [4], [9]. A *machine* is any total computable function. The complexity of a string $x \in \{0,1\}^*$ with respect to a machine ζ is defined as

$$C_\zeta(x) = \min_p \{l(p) : \zeta(p) = x\},$$

where p ranges over all partial functions (minimum over empty set is defined as ∞). There exists such a machine ζ that $C_\zeta(x) \leq C_{\zeta'}(x) + c_{\zeta'}$ for any x and any machine ζ' (the constant $c_{\zeta'}$ depends on ζ' but not on x). Fix any such ζ and define the *Kolmogorov complexity* of a string $x \in \{0,1\}^*$ as

$$C(x) := C_\zeta(x).$$

Clearly, $C(x) \leq |x| + b$ for any x and for some b depending only on ζ. A string is called c-incompressible if $C(x) \geq |x| - c$. Obviously, any data compressor can not compresses many c-incompressible strings, for any c. However, highly compressible strings (that is, strings with Kolmogorov complexity low relatively to their length) might be expected to be compressed well by some sensible data compressor. The following lemma shows that it can not be always the case, no matter what we mean by "relatively low".

The proof of this lemma is followed by the proof of Theorem 1.

Lemma 1. *For any data compressor ψ and any total recursive function $\gamma : \mathbb{N} \to \mathbb{N}$ such that γ goes monotonically to infinity there exists a binary string x such that $C(x) \leq \gamma(|x|)$ and $|\psi(x)| \geq |x|$.*

Proof. Suppose the contrary, i.e. that there exist an data compressor ψ and some function $\gamma : \mathbb{N} \to \mathbb{N}$ monotonically increasing to infinity such that for any string x if $C(x) \leq \gamma(|x|)$ then $\psi(x) < |x|$. Let T be the set of all strings which are not compressed by ψ

$$T := \{x : |\psi(x)| \geq |x|\}.$$

Define the function τ on the set T as follows: $\tau(x)$ is the number of the element x in T

$$\tau(x) := \#\{x' \in T : x' \leq x\}$$

for each $x \in T$. Obviously, the set T is infinite. Moreover, $\tau(x) \leq x$ for any $x \in T$ (recall that we identify $\{0,1\}^*$ and $\mathbb{N}$ via lexicographical ordering). Observe that τ is a total recursive function on T and onto $\mathbb{N}$. Thus $\tau^{-1} : \mathbb{N} \to \{0,1\}^*$ is a total recursive function on $\mathbb{N}$. Thus, for any $x \in T$,

$$C(\tau(x)) \geq C(\tau^{-1}(\tau(x)) - c = C(x) - c > \gamma(|x|) - c, \tag{1}$$

for constant c depending only on τ, where the first inequality follows from computability of τ^{-1} and the last from the definition of T.

It is a well-known result (see e.g. [4], Theorem 2.3.1) that for any partial function δ that goes monotonically to infinity there is $x \in \{0,1\}^*$ such that $C(x) \leq \delta(|x|)$. In particular, allowing $\delta(|x|) = \gamma(|x|) - 2c$, we conclude that there exist such $x \in T$ that

$$C(\tau(x)) \leq \gamma(|\tau(x)|) - 2c \leq \gamma(|x|) - 2c,$$

which contradicts (1). □

Proof of Theorem 1. Suppose the contrary, that is that there exists such a computable predictor φ and a total function $\beta : \mathbb{N} \to \mathbb{N}$ such that for any labelling function η, and any $n > \beta(l(\eta))$ we have

$$P\{x : \varphi(x_1, y_1, \ldots, x_n, y_n, x) \neq \eta(x)\} \leq 0.05,$$

for some $x_i \in X_{t(\eta)}$, $y_i = \eta(x_i)$, $i \in \mathbb{N}$, where P is the uniform distribution on $X_{t(\eta)}$.

Not restricting generality we can assume that β is strictly increasing. Define the (total) function $\beta^{-1}(n) := \max\{m \in \mathbb{N} : \beta(m) \leq n\}$. Define $\varepsilon := 0.05$. Construct the data compressor ψ as follows. For each $y \in \{0,1\}^*$ define $m := |y|$, $t := \ulcorner \log m \urcorner$. Generate (lexicographically) first m strings of

length t and denote them by x_i, $1 \le i \le m$. Define the labelling function η_y as follows: $t(\eta_y) = t$ and $\eta_y(x_i) = y^i$, $1 \le i \le m$. Clearly, $C(\eta_y) \le C(y) + c$, where c is some universal constant capturing the above description.

Let $n := \sqrt{m}$. Next we run the predictor φ on all possible tuples $\mathbf{x} = (x_1, \ldots, x_n) \in {X_t}^n$ and each time count errors that φ makes on all elements of X_t:

$$E(\mathbf{x}) := \{x \in X_t : \varphi(x_1, y^1, \ldots, x_n, y^n, x) \ne \eta_y(x)\}.$$

If $|E(\mathbf{x})| > \varepsilon m$ for each $\mathbf{x} \in X_t$ then $\psi(y) := 0y$.

Otherwise proceed as follows. Fix some tuple $\mathbf{x} = (x'_1, \ldots, x'_n)$ such that $|E(\mathbf{x})| \le \varepsilon m$, and let $H := \{x'_1, \ldots, x'_n\}$ be the unordered tuple $\mathbf{x}$. Define

$$\kappa^i := \begin{cases} e_0 & x_i \in E(\mathbf{x}) \backslash H, y^i = 0 \\ e_1 & x_i \in E(\mathbf{x}) \backslash H, y^i = 1 \\ c_0 & x_i \in H, y^i = 0 \\ c_1 & x_i \in H, y^i = 1 \\ * & \text{otherwise} \end{cases}$$

for $1 \le i \le m$. Thus, each κ^i is a member of a five-letter alphabet (a five-element set) $\{e_0, e_1, c_0, c_1, *\}$. Denote the string $\kappa^1 \ldots \kappa^m$ by K.

Observe that the string K, the predictor φ and the order of $(x'_1, \ldots, x'_n)$ (which is not contained in K) are sufficient to restore the string y. Furthermore, the n-tuple $(x'_1, \ldots, x'_n)$ can be obtained from H (the un-ordered tuple) by the appropriate permutation; let r be the number of this permutation in some fixed ordering of all $n!$ such permutations. Using Stirling's formula, we have $|r| \le 2n \log n$; moreover, to encode r with some self-delimiting code we need not more than $4n \log n$ symbols (for $n > 3$). Denote such encoding of r by ρ.

Next, as there are $(1 - \varepsilon - \frac{1}{\sqrt{m}})m$ symbols $*$ in the m-element string K, it can be encoded by some simple binary code σ in such a way that

$$|\sigma(K)| \le \frac{1}{2}m + 7(\varepsilon m + n). \tag{2}$$

Indeed, construct σ as follows. First replace all occurrences of the string $**$ with 0. Encode the rest of the symbols with any fixed 4-bit encoding such that the code of each letter starts with 1. Clearly, $\sigma(K)$ is uniquely decodable. Moreover, it is easy to check that (2) is satisfied, as there are not less than $\frac{1}{2}(m - 2(\varepsilon m + n))$ occurrences of the string $**$. We also need to write m in a self-delimiting way (denote it by s); clearly, $|s| \le 2 \log m$.

Finally, $\psi(\bar{y}) = 1\rho s\sigma(K)$ and $|\psi(y)| \leq |\bar{y}|$, for $m > 2^{10}$. Thus, ψ compresses any $\bar{y}$ such that $n > \beta(C(\eta_y))$; i.e. such that $\sqrt{m} > \beta(C(\eta_y)) \geq \beta(C(y) + c)$. This contradicts Lemma 1 with $\gamma(k) := \beta^{-1}(\sqrt{k} - c)$. □

4 On tightness of the negative results

In this section we discuss how tight are the conditions of the statements and to what extend they depend on the definitions.

Let us consider a question whether there exist any (not necessarily computable) sample-complexity function

$$\mathcal{N}_\varphi(k, \delta, \varepsilon) := \sup_{\eta: l(\eta) \leq k} N(\varphi, \eta, \delta, \varepsilon),$$

at least for some predictor φ, or it is always infinity from some k on.

Proposition 2. *There exist a predictor φ such that $\mathcal{N}_\varphi(k, \delta, \varepsilon) < \infty$ for any $\varepsilon, \delta > 0$ and any $k \in \mathbb{N}$.*

Proof. Clearly, $C(\eta) \geq C(t_\eta)$. Moreover, $\liminf_{t\to\infty} C(t) = \infty$ so that

$$\max\{t_\eta : l(\eta) \leq k\} < \infty$$

for any k. It follows that the "pointwise" predictor

$$\varphi(x_1, y_1, \ldots, x_n, y_n, x) = \begin{cases} y_i & \text{if } x = x_i, 1 \leq i \leq n \\ 0 & x \notin \{x_1, \ldots, x_n\} \end{cases} \tag{3}$$

satisfies the conditions of the proposition. □

It can be argued that probably this statement is due to our definition of a labelling function. Next we will discuss some other variants of this definition.

First, observe that if we define a labelling function as any total function on $\{0,1\}^*$ then some labelling functions will not approximate any real function; for example such is the function η_+ which counts bitwise sum of its input: $\eta_+(x) := \sum_{i=1}^{|x|} x_i \mod 2$. That is why we require a labelling function to be defined only on X_t for some t.

Another way to define a labelling function (which perhaps makes labelling functions most close to real functions) is as a function which accepts any *infinite* binary string. Let us call an *i-labelling function* any total recursive

function $\eta : \{0,1\}^\infty \to \{0,1\}$. That is, η is computable on a Turing machine with an input tape on which one way infinite input is written, an output tape and possibly some working tapes. The program η is required to halt on any input. The next proposition shows that even if we consider such definition the situation does not change. The definition of a labelling function η in which it accepts only finite strings is chosen in order to stay within conventional computability theory.

Lemma 2. *For any i-labelling function η there exist $n_\eta \in \mathbb{N}$ such that η does not scan its input tape further position n_η. In particular, $\eta(x) = \eta(x')$ as soon as $x_i = x'_i$ for any $i \leq n_\eta$.*

Proof. For any $x \in \{0,1\}^*$ the program η does not scan its tape further some position $n(x)$ (otherwise η does not halt on x). For any $\chi \in \{0,1\}^\infty$ denote by $n_\eta(\chi)$ the maximal $n \in \mathbb{N}$ such that η scans the input tape up to the position n on the input χ.

Suppose that $\sup_{\chi\in\{0,1\}^\infty} n_\eta(\chi) = \infty$, i.e. that the proposition is false. Define x^0 to be the empty string. Furthermore, let

$$x^i = \begin{cases} 0 & \sup_{\chi\in\{0,1\}^\infty} n_\eta(x^1, \ldots, x^{i-1}\chi) = \infty \\ 1 & \text{otherwise} \end{cases}$$

By our assumption, x_i is defined for each $i \in \mathbb{N}$. Moreover, it easy to check that η never stops on the input string $x_1 x_2 \ldots$. □

Besides, it is easy to check that the number n_η is computable.

Finally, it can be easily verified that Proposition 2 holds true if we consider i-labelling functions instead of labelling functions, constructing the required predictor based on the nearest neighbour predictor.

Proposition 3. *There exist a predictor φ such that ${}^i\mathcal{N}_\varphi(k, \delta, \varepsilon) < \infty$ for any $\varepsilon, \delta > 0$ and any $k \in \mathbb{N}$, where ${}^i\mathcal{N}$ is defined as $\mathcal{N}$ with labelling functions replaced by i-labelling functions.*

Proof. Indeed, it suffices to replace the "pointwise" predictor in the proof of Proposition 2 by the following predictor φ, which assigns to the object x the label of that object among $x_1, \ldots, x_n$ with whom x has longest mutual prefix: $\varphi(x_1, y_1, \ldots, x_n, y_n, x) := y_k$, where

$$k := \operatorname{argmax}_{1\leq m\leq n}\{\max\{i \in \mathbb{N} : x^1 \ldots x^i = x^1_m \ldots x^i_m\}\};$$

to avoid infinite run in case of ties, φ considers only first (say) n digits of x_i and break ties in favour of the lowest index. □

References

[1] L. Devroye, L. Györfi, G. Lugosi, A probabilistic theory of pattern recognition. New York: Springer, 1996.

[2] S. Jain, D. Osherson, J. Royer, and A. Sharma. Systems That Learn: An Introduction to Learning Theory, 2nd edition. The MIT Press, Cambridge, MA, 1999.

[3] M. Kearns and U. Vazirani An Introduction to Computational Learning Theory The MIT Press, Cambridge, Massachusetts, 1994.

[4] M. Li, P. Vitányi. An introduction to Kolmogorov complexity and its applications. Second edition, Springer, 1997.

[5] W. Menzel, F. Stephan. Inductive versus approximative learning. In: Perspectives of Adaptivity and learning, edited by R. Kuehn et al., pp. 187–209, Springer, 2003.

[6] H. Rogers. Theory of recursive functions and effective computability, McGraw-Hill Book Company, 1967.

[7] V. Vapnik, and A. Chervonenkis. Theory of Pattern Recognition. Nauka, Moscow, 1974

[8] V. Vapnik, Statistical Learning Theory: New York etc.: John Wiley & Sons, Inc. 1998

[9] A.K. Zvonkin and L.A. Levin. The complexity of finite objects and the development of the concepts of information and randomness by means of the theory of algorithms. Russian Math. Surveys, 25(6), pp 83–124, 1970.