SCIENTIFIC REPORTS
nature research

OPEN

Generic predictions of output probability based on complexities of inputs and outputs

Kamaludin Dingle[1,2], Guillermo Valle Pérez[2] & Ard A. Louis[2*]

For a broad class of input-output maps, arguments based on the coding theorem from algorithmic information theory (AIT) predict that simple (low Kolmogorov complexity) outputs are exponentially more likely to occur upon uniform random sampling of inputs than complex outputs are. Here, we derive probability bounds that are based on the complexities of the inputs as well as the outputs, rather than just on the complexities of the outputs. The more that outputs deviate from the coding theorem bound, the lower the complexity of their inputs. Since the number of low complexity inputs is limited, this behaviour leads to an effective lower bound on the probability. Our new bounds are tested for an RNA sequence to structure map, a finite state transducer and a perceptron. The success of these new methods opens avenues for AIT to be more widely used.

Deep links between physics and theories of computation[1,2] are being increasingly exploited to uncover new fundamental physics and to provide novel insights into theories of computation. For example, advances in understanding quantum entanglement are often expressed in sophisticated information theoretic language, while providing new results in computational complexity theory such as polynomial time algorithms for integer factorization[3]. These connections are typically expressed in terms of Shannon information, with its natural analogy with thermodynamic entropy.

There is, however, another branch of information theory, called algorithmic information theory (AIT)[4], which is concerned with the information content of individual objects. It has been much less applied in physics (although notable exceptions occur, see[5] for a recent overview). Reasons for this relative lack of attention include that AIT's central concept, the Kolmogorov complexity $K_U(x)$ of a string x, defined as the length of the shortest program that generates x on an optimal reference universal Turing machine (UTM) U, is formally uncomputable due to its link to the famous halting problem of UTMs[6] — see[7] for technical details. Moreover, many important results, such as the invariance theorem which states that for two UTMs U and W, the Kolmogorov complexities $K_U(x) = K_W(x) + \mathcal{O}(1)$ are equivalent, hold asymptotically up to $\mathcal{O}(1)$ terms that are independent of x, but not always well understood, and therefore hard to control.

Another reason applications of AIT to many practical problems have been hindered can be understood in terms of hierarchies of computing power. For example, one of the oldest such categorisations, the Chomsky hierarchy[8], ranks automata into four different classes, of which the UTMs are the most powerful, and finite state machines (FSMs) are the least. Many key results in AIT are derived by exploiting the power of UTMs. Interestingly, if physical processes can be mapped onto UTMs, then certain properties can be shown to be uncomputable[9,10]. However, many problems in physics are fully computable, and therefore lower on the Chomsky hierarchy than UTMs. For example, finite Markov processes are equivalent to FSMs, and RNA secondary structure (SS) folding algorithms can be recast as context-free grammars, the second level in the hierarchy. Thus, an important cluster of questions for applications of AIT revolve around extending its methods to processes lower in computational power than UTMs.

To explore ways of moving beyond these limitations and towards practical applications, we consider here one of the most iconic results of AIT, namely Levin's coding theorem[11] (see also the work of Solomonoff[12] who had a version of the a priori distribution), which predicts that upon randomly chosen programs (e.g. a binary string program constructed by coin flips), the probability $P_U(x)$ that a universal Turing machine (UTM) generates output x can be bounded as $2^{-K(x)} \leq P(x) \leq 2^{-K(x)+\mathcal{O}(1)}$. Technically, because we require the programs of the Turing

[1]Centre for Applied Mathematics and Bioinformatics, Department of Mathematics and Natural Sciences, Gulf University for Science and Technology, Hawally, 32093, Kuwait. [2]Rudolf Peierls Centre for Theoretical Physics, University of Oxford, Parks Road, Oxford, OX1 3PU, United Kingdom. *email: ard.louis@physics.ox.ac.uk

machine to be prefix-free, we restrict ourselves to what are called prefix Turing machines. Given this profound prediction of a general exponential bias towards simplicity (low Kolmogorov complexity) one might have expected widespread study and applications in science and engineering. This has not been the case because the theorem unfortunately suffers from the general issues of AIT described above.

Nevertheless, it has recently been shown[13,14] that a related exponential bias towards low complexity outputs obtains for a range of non-universal input-output maps f: $I \to O$ that are lower on the Chomsky hierarchy than UTMs.

In particular, an upper bound on the probability $P(x)$ that an output obtains upon uniform random sampling of inputs,

$$P(x) \leq 2^{-a\widetilde{K}(x)-b} \tag{1}$$

was recently derived[13] using a computable approximation $\widetilde{K}(x)$ to the Kolmogorov complexity of x, typically calculated using lossless compression techniques. Here a and b are constants that are independent of x and which can often be determined from some basic information about the map. The so-called *simplicity bias bound* (1) holds for computable maps f where the number of inputs N_I is much greater than the number of outputs N_O and the map f is simple, meaning that asymptotically $K(f) + K(n) \ll K(x) + \mathcal{O}(1)$ for a typical output x, where n specifies the size of N_I, e.g. $N_I = 2^n$. It is important to distinguish the map f (which we assume is simple) and the initial conditions[15], i.e. a program for x (which may or may not be simple).

Equation (1) typically works better for larger N_I and N_O. Approximating the true Kolmogorov complexity also means that the bound shouldn't work for maps where a significant fraction of outputs have complexities that are not qualitatively captured by compression based approximations. For example many pseudo random-number generators are designed to produce outputs that appear to be complex when measured by compression or other types of Kolmogorov complexity approximations. Yet these outputs must have low $K(x)$ because they are generated by relatively simple algorithms with short descriptions. Nevertheless, it has been shown that the bound (1) works remarkably well for a wide class of input-output maps, ranging from the sequence to RNA secondary structure map, to systems of coupled ordinary differential equations, to a stochastic financial trading model, to the parameter-function map for several classes of deep neural networks[13,16,17].

The simplicity bias bound (1) predicts that high $P(x)$ outputs will be simple, and that complex outputs will have a low $P(x)$. But, in sharp contrast to the full AIT coding theorem, it doesn't have a lower bound, allowing low $\widetilde{K}(x)$ outputs with low $P(x)$ that are far from the bound. Indeed, this behaviour is generically observed for many (non-universal) maps[13,16] (see also Fig. 1), but should not be the case for UTMs that obey the full coding theorem. Understanding the behaviour of outputs far from the bound should shed light on fundamental differences between UTMs and maps with less computational power that are lower on the Chomsky hierarchy, and may open up avenues for wider applications of AIT in physics.

Results

With this challenge in mind, we take an approach that contrasts with the traditional coding theorem of AIT or with the simplicity bias bound, which only consider the complexity of the outputs. Instead, we derive bounds that also take into account the complexity of the inputs that generate a particular output x. While this approach is not possible for UTMs, since the halting problem means one cannot enumerate all inputs[4], and so averages over input complexity cannot be calculated, it can be achieved for non-UTM maps. Among our main results, we show that the further outputs are from the simplicity bias bound (1), the lower the complexity of the set of inputs. Since, by simple counting arguments, most strings are complex[4], the cumulative probability of outputs far from the bound is therefore limited. We also show that by combining the complexities of the output with that of the inputs, we can obtain better bounds on and estimates of $P(x)$.

Whether such bounds nevertheless have real predictive power needs to be tested empirically. Because input based bounds typically need exhaustive sampling, full testing is only possible for smaller systems, which restricts us here to maps where finite size effects may still play a role[13]. We test our bounds on three systems, the famous RNA sequence to secondary structure map (which falls into the context-free class in the Chomsky hierarchy), here for a relatively small size with length $L = 15$ sequences, a finite state transducer (FST), a very simple input-output map that is lowest on the Chomsky hierarchy[8], with length $L = 30$ binary inputs, and finally the parameter-function map[16,17] of a perceptron[18] with discretized weights to allow complexities of inputs to be calculated. The preceptron plays a key role in deep learning neural network architectures[19]. Nevertheless, as can be seen in Fig. 1(a–c) all three maps exhibit simplicity bias predicted by Eq (1), even if they are relatively small. In ref. [13], much cleaner simplicity bias behaviour can be observed for larger RNA maps, but these are too big to exhaustively sample inputs. Similarly, cleaner simplicity bias behaviour occurs for the undiscretised perceptron[17], but then it is hard to analyse the complexity of the inputs. Figure 1(a–c) shows that the complexity of the input strings that generate each output x decreases for further distances from the simplicity bias bound. This is the kind of phenomenon that the we will attempt to explain.

To study input based bounds, consider a map f: $I \to O$ between N_I inputs and N_O outputs that satisfies the requirements for simplicity bias[13]. Let $f(p) = x$, where p is some input program $p \in I$ producing output $x \in O$. For simplicity let $p \in \{0, 1\}^n$, so that all inputs have length n and $N_I = 2^n$ (this restriction can be relaxed later). Define $f^{-1}(x)$ to be the set of all the inputs that map to x, so that the probability that x obtains upon sampling inputs uniformly at random is

$$P(x) = \frac{|f^{-1}(x)|}{2^n} \tag{2}$$

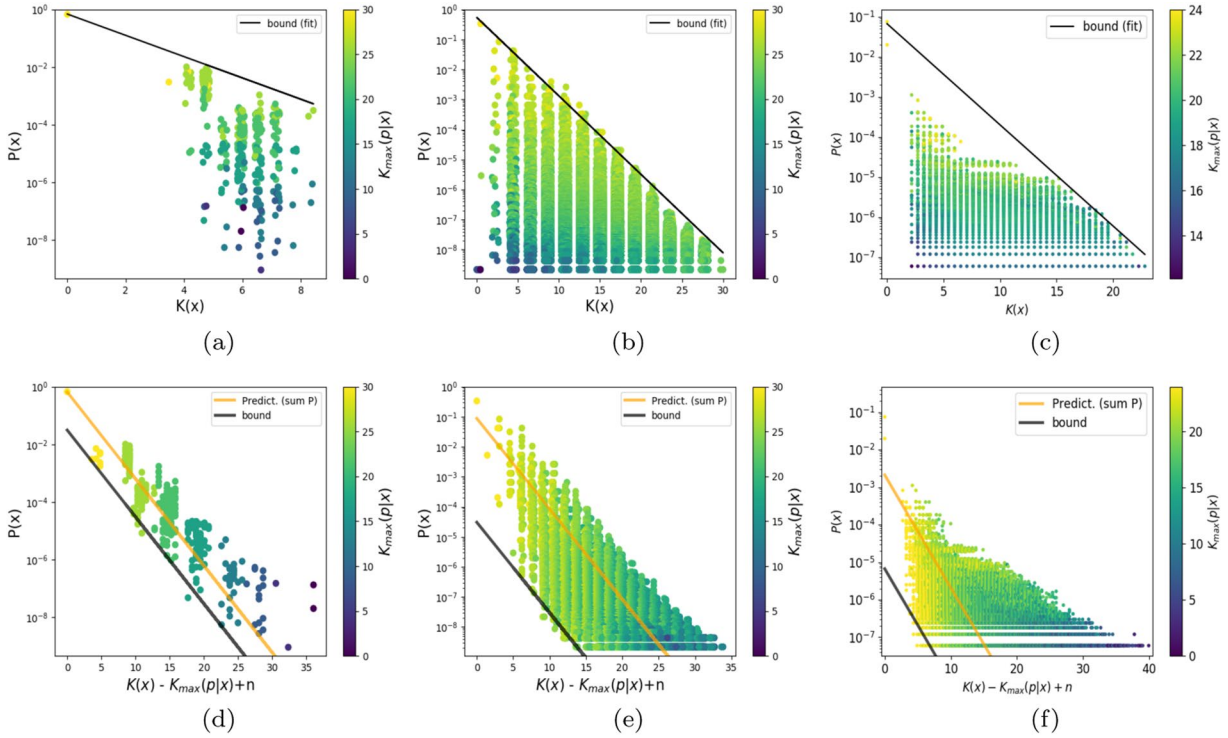

Figure 1. The probability $P(x)$ that a particular output arises upon random sampling of inputs versus output complexity $\widetilde{K}(x)$ shows clear simplicity bias for: (**a**) A length $L = 15$ RNA sequence to SS mapping, (**b**) An FST, sampled over all 2^{30} binary inputs of length 30, and (**c**) A 7-input perceptron with weights discretised to 3 bits. The black solid line is the simplicity bias bound (1) (with a and b fit). For all these maps high complexity outputs occur with low probability. The outputs are colour coded by the maximum complexity $K_{\max}(p|n)$ of the set of inputs mapping to output x. Outputs further from the bound have lower input complexities. (**d**) length $L = 15$ RNA, (**e**) the FST and (**d**) the perceptron, show the data plotted for the lower bound (8) (black line) with only the intercept fit to the data, the slope is a prediction. The orange line is using Eq (8) with a normalised probability for a parameter free predictor. Including the complexity of the input through $K_{\max}(p|n)$ reduces the spread in the data, and so provides more predictive power than $K(x)$ alone.

Any arbitrary input p can be described using the following $\mathcal{O}(1)$ procedure[13]: Assuming f and n are given, first enumerate all 2^n inputs and map them to outputs using f. The index of a specific input p within the set $f^{-1}(x)$ can be described using at most $\log_2(|f^{-1}(x)|)$ bits. In other words, this procedure identifies each input by first finding the output x it maps to, and then finding its label within the set $f^{-1}(x)$. Given f and n, an output $x = f(p)$ can be described using $K(x|f, n) + \mathcal{O}(1)$ bits[13]. Thus, the Kolmogorov complexity of p, given f and n can be bounded as:

$$K(p|f, n) \leq K(x|f, n) + \log_2(|f^{-1}(x)|) + \mathcal{O}(1). \tag{3}$$

We note that this bound holds in principle for all p, but that it is tightest for $K_{\max}(p|x) \equiv \max_p\{K(p|f, n)\}$ for $p \in f^{-1}(x)$. More generally, we can expect these bounds to be fairly tight for the maximum complexity $K_{\max}(p|f, n)$ of inputs due to the following argument. First note that

$$K_{\max}(p|f, n) \geq \log_2(|f^{-1}(x)|) + \mathcal{O}(1) \tag{4}$$

because any set of $|f^{-1}(x)|$ different elements must have strings of at least this complexity. Next,

$$K(x|f, n) \leq K(p|f, n) + \mathcal{O}(1) \tag{5}$$

because each p can be used to generate x. Therefore:

$$\max(K(x|f, n), \log_2(|f^{-1}(x)|)) \leq K_{\max}(p|f, n) + \mathcal{O}(1), \tag{6}$$

so the bound (3) cannot be too weak. In the worst case scenario, where $K_{\max}(p|n) \approx \log_2(|f^{-1}(x)|) \approx K(x|f, n)$, the right hand side of the bound (3) is approximately twice the left hand side (up to additive $\mathcal{O}(1)$ terms). It is tighter if either $K(x|f, n)$ is small, or if $K(x|f, n)$ is big relative to $\log_2(|f^{-1}(x)|)$. As is often the case for AIT predictions, the stronger the constraint/prediction, the more likely it is to be observed in practice, because, for example, the $\mathcal{O}(1)$ terms are less likely to drown out the effects.

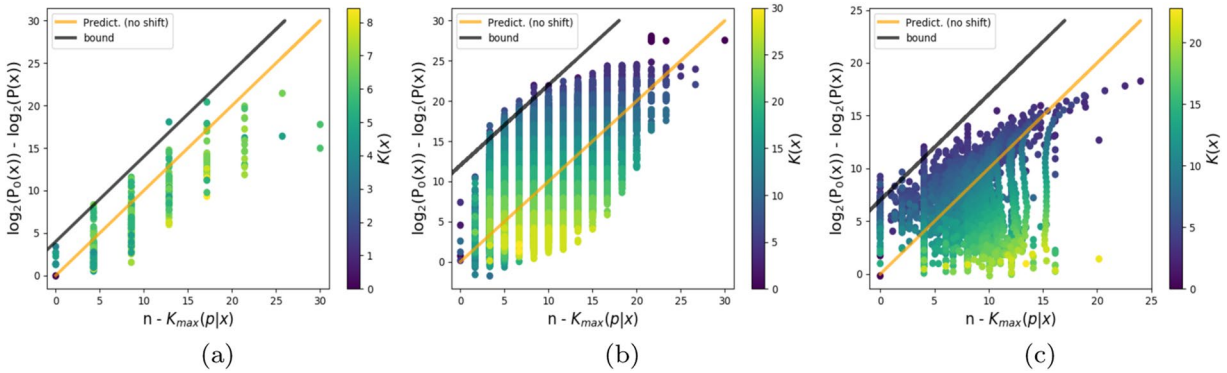

Figure 2. Deviation of $P(x)$ from the simplicity bias upper bound (1) increases with increasing randomness deficit $\delta_{\max}(x) = n - K_{\max}(p|n)$ for (**a**) $L = 15$ RNA, (**b**) $L = 30$ FST, (**c**) perceptron with weights discretised to 4 bits. For the perceptron, all functions with the same $P(x)$ and $K(x)$ are averaged together to reduce scatter. Points are colour coded by output complexity $K(x)$. For the upper bound (9) (black line) we fit the intercept, but the slope is a prediction, if we treat it as a normalised probability we obtain the orange line which is a direct prediction with no free parameters.

By combining with Eq. (2), the bound (3) can be rewritten in two complementary ways. Firstly, a lower bound on $P(x)$ can be derived of the form:

$$P(x) \geq 2^{-K(x|f,n)-[n-K(p|f,n)]+\mathcal{O}(1)} \tag{7}$$

$\forall\, p \in f^{-1}(x)$ which complements the simplicity bias upper bound (1). This bound is tightest for $K_{\max}(p|n)$.

In ref. [13] it was shown that $P(x) \leq 2^{-K(x|f,n)+\mathcal{O}(1)}$ by using a similar counting argument to that used above, together with a Shannon-Fano-Elias code procedure. Similar results can be found in standard works[4,20]. A key step is to move from the conditional complexity to one that is independent of the map and of n. If f is simple, then the explicit dependence on n and f can be removed by noting that since $K(x) \leq K(x|f, n) + K(f) + K(n) + \mathcal{O}(1)$, and $K(x|f, n) \leq K(x) + \mathcal{O}(1)$ then $K(x|f, n) \approx K(x) + \mathcal{O}(1)$. In Eq. (1) this is further approximated as $K(x|f, n) + \mathcal{O}(1) \approx a\widetilde{K}(x) + b$, leading to a practically useable upper bound. The same argument can be used to remove explicit dependence on n and f for $K(p|f, n)$.

If we define a maximum randomness deficit $\delta_{\max}(x) = n - K_{\max}(p|n)$, then this tightest version of bound (7) can be written in a simpler form as

$$P(x) \geq 2^{-a\widetilde{K}(x)+b-\delta_{\max}(x)+\mathcal{O}(1)} \tag{8}$$

In Fig. 1(d–f) we plot this lower bound for all three maps studied. Throughout the paper, we use a scaled complexity measure, which ensures that $\widetilde{K}(x)$ ranges between ≈ 0 and $\approx n$ bits, for strings of length n, as expected for Kolmogorov complexity. See Methods for more details.

When comparing the data in Fig. 1(d–f) to Fig. 1(a–c), it is clear that including the input complexities reduces the spread in the data for RNA and the FST, although for the perceptron model the difference is less pronounced. This success suggests using the bound (8) as a predictor $P(x) \approx 2^{-K(x|f,n)-\delta_{\max}(x)}$, with the additional constraint that $\sum_x P(x) = 1$ to normalise it. As can be seen in Fig. 1(d–f), this simple procedure works reasonably well, showing that the input complexity provides additional predictive power to estimate $P(x)$ from some very generic properties of the inputs and outputs.

A second, complimentary way that bound (3) can be expressed is in terms of how far $P(x)$ differs from the simplicity bias bound (1):

$$[\log_2(P_0(x)) - \log_2(P(x))] \leq [n - K(p|f, n)] + \mathcal{O}(1) \tag{9}$$

where $P_0(x) = 2^{-K(x|f,n)} \approx 2^{-a\widetilde{K}(x)+b}$ is the upper bound (1) shown in Fig. 1(a–c).

For a random input p, with high probability we expect $K(p|f, n) = n + \mathcal{O}(1)$[4]. Thus, Eqs. (7) and (9) immediately imply that large deviations from the simplicity bias bound (1) are only possible with highly non-random inputs with a large randomness deficit $\delta_{\max}(x)$.

In Fig. 2(a–c) we directly examine bound (9), showing explicitly the prediction that a drop of probability $P(x)$ by Δ bits from the simplicity bias bound (1) corresponds to a Δ bit randomness deficit in the set of inputs.

Simple counting arguments can be used to show that the number of non-random inputs is a small fraction of the total number of inputs[21]. For example, for binary strings of length n, with $N_I = 2^n$, the number of inputs with complexity $K = n - \delta$ is approximately $2^{-\delta}N_I$. If we define a set $\mathcal{D}(f)$ of all outputs x_i that satisfy $(\log_2(P_0(x_i)) - \log_2(P_{(x_i)})) \geq \Delta$, i.e. the set of all outputs for which $\log_2 P(x)$ is at least Δ bits below the simplicity bias bound (1), then this counting argument leads to the following cumulative bound:

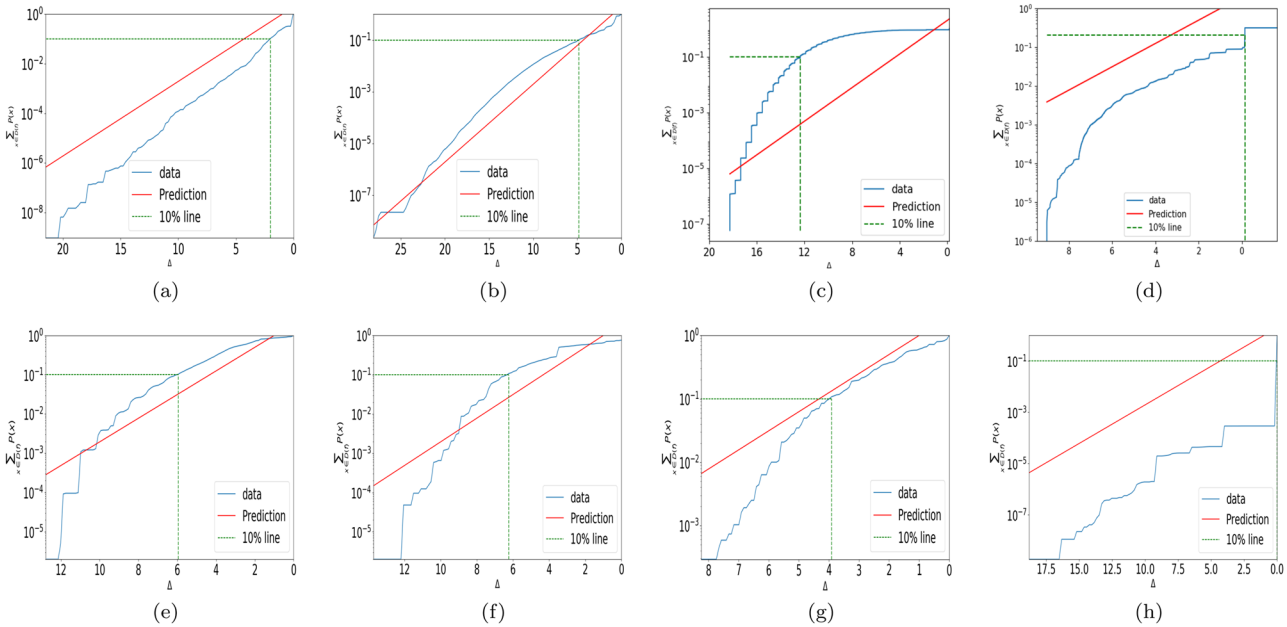

Figure 3. The cumulative probability versus the distance from the bound Δ correlates with the the cumulative bound (10) (red line) for (**a**) L = 15 RNA and (**b**) L = 30 FST (**c**) Perceptron. (**d**) fully connected 2 layer neural network from[16], (**e**) coarse-grained ordinary differential equation map from[13], which describes a circadian rhythm model from[40], (**f**) Ornstein-Uhlenbeck financial model from[13], (**g**) L-systems from[13], (**h**) simple matrix map from[13]. The solid red line is the prediction $2^{-\Delta+1}$ from Eq. (10), the dashed line denotes 10% cumulative probability.

$$\sum_{x\in\mathcal{D}(f)} P(x) \leq 2^{-\Delta+1+\mathcal{O}(1)} \tag{10}$$

which predicts that, upon randomly sampling inputs, most of the probability weight is for outputs with $P(x)$ relatively close to the upper bound. There may be many outputs that are far from the bound, but their cumulative probability drops off exponentially the further they are from the bound because the number of simple inputs is exponentially limited. Note that this argument is for a cumulative probability over all inputs. It does not predict that for a given complexity $K(x)$, that the outputs should all be near the bound. In that sense this lower bound is not like that of the original coding theorem which holds for any output x. See the Supplementary Information for an alternative derivation of this bound.

Bound (10) does not need an exhaustive enumeration to be tested. In Fig. 3 we show this bound for a series of different maps, including many maps from[13]. The cumulative probability weight scales roughly as expected, implying that most of the probability weight is relatively close to the bound (at least on a log scale).

What is the physical nature of these low complexity, low probability outputs that occur far from the bound? They must arise in one way or another from the lower computational power of these maps, since they don't occur in the full AIT coding theory. Low complexity, low probability outputs correspond to output patterns which are simple, but which the given computable map is not good at generating.

In RNA it is easy to construct outputs which are simple but will have low probability. Compare two $L = 15$ structures $S_1 = ((.(.(...).).))$. and $S_2 = .(((.(...)).))$., which are both symmetric and thus have a relatively low complexity $K(S_1) = K(S_2) = 21.4$. Nevertheless they have a significant difference in probability, $P(S_2)/P(S_1) \approx 560$ because S_1 has several single bonds, which is much harder to make according to the biophysics of RNA. Only specially ordered input sequences can make S_1, in other words they are simple, with $K_{\max}(p|n) = 8.6$. By contrast, the inputs of S_2 are much higher at $K_{\max}(p|n) = 21.4$ because they need to be constrained less to produce this structure. This example illustrates how the system specific details of the RNA map can unfavourably bias away from some outputs due to a system specific constraint.

Similar examples of system specific constraint for the FST and perceptron can be found in the SI. We hypothesise that such low complexity, low probability structures highlight specific non-universal aspects of the maps, and extra information (in the form of a reduced set of inputs) are needed to generate such structures.

Discussion

In conclusion, it is striking that bounds based simply on the complexity of the inputs and outputs can make powerful and general predictions for such a wide range of systems. Although the arguments used to derive them suffer from the well known problems – e.g. the presence of uncomputable Kolmogorov complexities and unknown $\mathcal{O}(1)$ terms – that have led to the general neglect of AIT in the physics literature, the bounds are undoubtably successful. It appears that, just as is found in other areas of physics, these relationships hold well outside of the asymptotic

regime where they can be proven to be correct. This practical success opens up the promise of using such AIT based techniques to derive other results for computable maps from across physics.

Many new questions arise. Can it be proven when the $\mathcal{O}(1)$ terms are relatively unimportant? Why do our rather simple approximations to $K(x)$ work? It would be interesting to find maps where these classical objections to the practical use of AIT are important. There may also be connections between our work and finite state complexity[22] or minimum description length[23] approaches. Progress in these domains should generate new fundamental understandings of the physics of information.

Methods

RNA sequence to secondary structure mapping. RNA is made of a linear sequence of 4 different kinds of nucleotides, so that there are $N_I = 4^L$ possible sequences for any particular length L. A versatile molecule, it can store information, as messenger RNA, or else perform catalytic or structural functions. For functional RNA, the three-dimensional (3D) structure plays an important role in its function. In spite of decades of research, it remains difficult to reliably predict the 3D structure from the sequence alone. However, there are fast and accurate algorithms to calculate the so-called secondary structure (SS) that determines which base binds to which base. Given a sequence, these methods typically minimize the Turner model[24] for the free-energy of a particular bonding pattern. The main contributions in the Turner model are the hydrogen bonding and stacking interactions between the nucleotides, as well as some entropic factors to take into account motifs such as loops. Fast algorithms based on dynamic programming allow for rapid calculations of these SS, and so this mapping from sequences to SS has been a popular model for many studies in biophysics.

In this context, we view it as an input-output map, from N_I input sequences to N_O output SS structures. This map has been extensively studied (see e.g.[25–33]) and provided profound insights into the biophysics of folding and evolution.

Here we use the popular Vienna package[26] to fold sequences to structures, with all parameters set to their default values (e.g. the temperature $T = 37°C$). We folded all $N_I = 4^{15} \approx 10^9$ sequences of length 15, into 346 different structures which were the free-energy minimum structures for those sequences. The number of sequences mapping to a structure is often called the *neutral set size*.

The structures can be abstracted in standard dot-bracket notation, where brackets denote bonds, and dots denote unbonded pairs. For example, ...((....)).... means that the first three bases are not bonded, the fourth and fifth are bonded, the sixth through ninth are unbonded, the tenth base is bonded to the fifth base, the eleventh base is bonded to the fourth base, and the final four bases are unbounded.

To estimate the complexity of an RNA SS, we first converted the dot-bracket representation of the structure into a binary string x, and then used the complexity estimator described below to estimate its complexity. To convert to binary strings, we replaced each dot with the bits 00, each left-bracket with the bits 10, and each right-bracket with 01. Thus an RNA SS of length n becomes a bit-string of length $2n$. As an example, the following $n = 15$ structure yields the displayed 30-bit string

$$..(((...)))....\ \rightarrow\ 000010101000000001010100000000$$

Because we are interested in exhaustive calculations, we are limited to rather small RNA sequence lengths. This means that finite-size effects may play an important role. In[13], we compared the simplicity bias bound (1) from the main text to longer sequences where only partial sampling can be achieved, and showed much clearer simplicity bias is evident in those systems.

Finite state transducer. Finite state transducers (FSTs) are a generalization of finite state machines that produce an output. They are defined by a finite set of states $\mathcal{S}$, finite input and output alphabets $\mathcal{I}$ and $\mathcal{O}$, and a transition function T: $\mathcal{S} \times \mathcal{I} \rightarrow \mathcal{S} \times \mathcal{O}$ defining, for each state, and input symbol, a next state, and output symbol. One also needs to define a distinguished state, $S_0 \in \mathcal{S}$, which will be the initial state, before any input symbol has been read. Given an *input sequence* of L input symbols, the system visits different states, and simultaneously produces an *output sequence* of L output symbols.

FST form a popular toy system for computable maps. They can express any computable function that requires only a finite number of memory, and the number of states in the FSTs offers a good parameter to control the complexity of the map. The class of machines we described above is also known as Mealy machines[34]. If one restricts the transition function to only depend on the current state, one obtains Moore machines[35]. If one considers the input sequence to a Moore machine to be stochastic, it immediately follows that its state sequence follows a Markov chain, and its output sequence is a Markov information source. Therefore, FSTs can be used to model many stochastic systems in nature and engineering, which can be described by finite-state Markov dynamics.

FSTs lie in the lowest class in the Chomsky hierarchy. However, they appear to be biased towards simple outputs in a manner similar to Levin's coding theorem. In particular, Zenil *et al.*[14] show evidence of this by correlating the probability of FSTs and UTMs producing particular outputs. More precisely, they sampled random FSTs with random inputs, and random UTMs with random inputs, and then compared the empirical frequencies with with individual output strings are obtained by both families, after many samples of machines and inputs. For both types of machines, simple strings were much more likely to be produced than complex strings.

We use randomly generated FSTs with 5 states. The FSTs are generated by uniformly sampling complete initially connected DFAs (where every state is reachable from the initial state, and the transition function is defined for every input) using the library FAdo[36], which uses the algorithm developed by Almeida *et al.*[37]. Output symbols are then added to each transition independently and with uniform probability. In our experiments, the inputs are binary strings and the outputs are binary strings of length $L = 30$. The outputs for the whole set of 2^L input strings are computed using the HFST library (https://hfst.github.io/). Not all FSTs show bias, but we have observed that

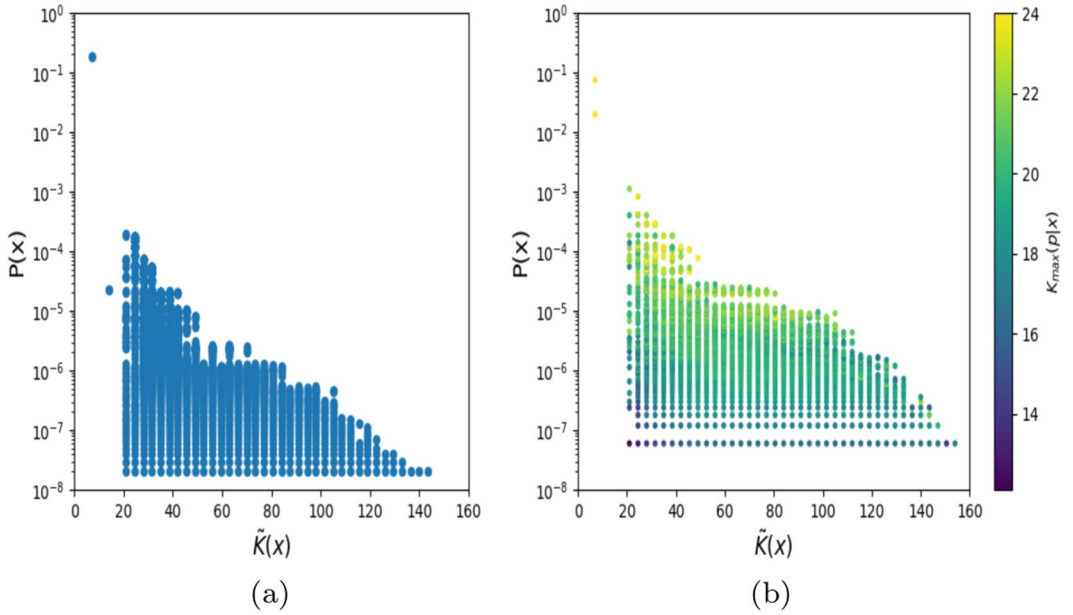

Figure 4. Methods: Probability versus complexity $\widetilde{K}(x)$ (measured here as $C_{LZ}(x)$ from Eq. (11) shows simplicity bias in the perceptron for (**a**) full continuous weights and (**b**) with discretised weights (as in Fig. 1 of the main text). Since weights and biases are real-valued in (**a**) it is not straightforward to measure the complexity of the inputs. It is, of course, possible to do so for the discretised weights of (**b**).

all those that show bias show simplicity bias, and have the same behavior as that shown in Fig. 1 for low complexity - low probability outputs.

We can see why some simple outputs will occur with low probability by considering system specific details of the FST. For an FST, an output of length n which is $n/2$ zeros followed by $n/2$ ones is clearly simple, but we find that it has a low probability. We can understand this intuitively as follows. Producing such a string requires the "counting" up to $n/2$ to know when to switch output, and counting requires a memory that grows with n, while FSTs have finite memory. We can also prove that, for instance, an FST that only produces such strings (for any n) is impossible. The set of possible strings that an FST can produce comprises a regular language, as constructed by using the output symbols at each transition as input symbols, giving us a non-deterministic finite automaton. Finally, using the pumping lemma[38], it is easy to see that this family of strings isn't a regular language.

Perceptron. The perceptron[18] is the simplest type of artificial neural network. It consists of a single linear layer, and a single output neuron with binary activation. Because modern deep neural network architectures are typically made of many layers of perceptrons, this simple system is important to study[17]. In this paper we use perceptrons with Boolean inputs and discretized weights. For inputs $x \in \{0, 1\}^n$, the discretized perceptron uses the following parametrized class of functions:

$$f_{w,b}(x) = \mathbf{1}(w \cdot x + b),$$

where $w \in \{-a, a+\delta, \ldots, a-\delta, a\}^n$ and $b \in \{-a, a+\delta, \ldots, a-\delta, a\}$ are the weight vector and bias term, which take values in a discrete lattice with $D := 2a/\delta + 1$ possible values per weight. We used $D = 2^k$, so that each weight can be represented by k bits, and $a = (2^k - 1)/2$, so that $\delta = 1$. Note that rescaling all the weights w and the bias b by the same fixed constant wouldn't change the family of functions.

To obtain the results in Fig. 1, for which $n = 7$, we represented the weights and bias with $k = 3$ bits. We exhausitvely enumerated all $2^{3(7+1)}$ possible values of the weights and vectors, and we counted how many times we obtained each possible Boolean function on the Boolean hypercube $\{0, 1\}^7$. The weight-bias pair was represented using $3 \times (7 + 1) = 24$ bits. A pair (w, b) is an input to the parameter-function map of the perceptron. The complexity of inputs to this map can therefore be approximated by computing the Lempel-Ziv complexity of the 24-bit representation of the pair (w, b).

In Fig. 4, we compare the simplicity bias of a perceptron with real-valued weights and bias, sampled from a standard Normal distribution, to the simplicity bias of the perceptron with discretized weights. We observe that both display similar simplicity bias, although the profile of the upper bound changes slightly.

For the perceptron we can also understand some simple examples of low complexity, low probability outputs. For example, the function with all 0s except a 1, for the inputs (1, 0, 0, 0, 0, 0, 0) and (0, 1, 0, 0, 0, 0, 0, 0) has a similar complexity to the function which only has 1s at the inputs (1, 0, 0, 0, 0, 0, 0, 0) and (0, 1, 1, 1, 1, 1, 1, 1). However, the latter has much lower probability. One can understand this because if we take the dot product of a random weight vector w with two different inputs x_1 and x_2, the results have correlation given $x_1 \cdot x_2/(||x_1||||x_2||)$. Therefore we expect the input (0, 1, 1, 1, 1, 1, 1) to be correlated to more other inputs, than (0, 1, 0, 0, 0, 0, 0), so that the probability of it having a different value than the majority of inputs (as is the case for the second function) is expected to be significantly lower.

Methods to estimate complexity $\tilde{K}(x)$. There is a much more extensive discussion of different ways to estimate the Kolmogorov complexity in the supplementary information of[13] and[16]. Here we use compression based measures, and as in these previous papers, we these are based on the 1976 Lempel Ziv (LZ) algorithm[39], but with some small changes:

$$C_{LZ}(x) = \begin{cases} \log_2(n), & x = 0^n \text{ or } 1^n \\ \log_2(n)[N_w(x_1 \ldots x_n) + N_w(x_n \ldots x_1)]/2, & \text{otherwise} \end{cases} \tag{11}$$

Here $N_w(x)$ is the number of code words found by the LZ algorithm. The reason for distinguishing 0^n and 1^n is merely an artefact of $N_w(x)$ which assigns complexity $K = 1$ to the string 0 or 1, but complexity 2 to 0^n or 1^n for $n \geq 2$, whereas the Kolmogorov complexity of such a trivial string actually scales as $\log_2(n)$, as one only needs to encode n. In this way we ensure that our $C_{LZ}(x)$ measure not only gives the correct behaviour for complex strings in the $\lim_{n\to\infty}$, but also the correct behaviour for the simplest strings. In addition to the $\log_2(n)$ correction, taking the mean of the complexity of the forward and reversed strings makes the measure more fine-grained, since it allows more values for the complexity of a string. Note that $C_{LZ}(x)$ can also be used for strings of larger alphabet sizes than just 0/1 binary alphabets.

To directly test the input based measures we typically need fairly small systems, where the LZ based measure above may show some anomalies (see also the supplementary information of[13] for a more detailed description). Thus, for such small systems, or when comparing different types and sizes of objects (e.g. RNA SS and RNA sequences) a slightly different scaling may be more appropriate, which not only accounts for the fact that $C_{LZ}(x) > n$ for strings of length n, but also the lower complexity limit may not be ~0, which it should be (see also the discussion in the supplementary information of[13]). Hence we use a different rescaling of the complexity measure

$$\widetilde{K}(x) = \log_2(N_O) \cdot \frac{C_{LZ}(x) - \min(C_{LZ}(x))}{\max(C_{LZ}(x)) - \min(C_{LZ}(x))} \tag{12}$$

which will now range between $0 \leq \widetilde{K}(x) \leq \log_2(N_O) = n$ if for example $N_O = 2^n$. For large objects, this different scaling will reduce to the simpler one, because $\max(C_{LZ}(x)) \gg \min(C_{LZ}(x))$.

We note that there is nothing fundamental about using LZ to generate approximations to true Kolmogorov complexity. Many other approximations could be used, and their merits may depend on the details of the problems involved. For further discussion of other complexity measures, see for example the supplementary information of refs. [13,16].

Data availability

The data that support the findings of this work are available from the corresponding authors upon request.

Received: 12 November 2019; Accepted: 16 February 2020;
Published online: 10 March 2020

References

1. Mezard, M. & Montanari, A. *Information, physics, and computation* (Oxford University Press, USA, 2009).
2. Moore, C. & Mertens, S. *The nature of computation* (OUP Oxford, 2011).
3. Shor, P. W. Algorithms for quantum computation: Discrete logarithms and factoring. In *Proceedings 35th annual symposium on foundations of computer science*, 124–134 (Ieee, 1994).
4. Li, M. & Vitanyi, P. *An introduction to Kolmogorov complexity and its applications* (Springer-Verlag New York Inc, 2008).
5. Devine, S. Algorithmic information theory: Review for physicists and natural scientists (2014).
6. Turing, A. M. On computable numbers, with an application to the entscheidungsproblem. *J. of Math* **58**, 5 (1936).
7. Shen, A., Uspensky, V. A. & Vereshchagin, N. *Kolmogorov complexity and algorithmic randomness*, vol. 220 (American Mathematical Soc., 2017).
8. Chomsky, N. Three models for the description of language. *Information Theory, IRE Transactions on* **2**, 113–124 (1956).
9. Lloyd, S. Quantum-mechanical computers and uncomputability. *Physical review letters* **71**, 943 (1993).
10. Cubitt, T. S., Perez-Garcia, D. & Wolf, M. M. Undecidability of the spectral gap. *Nature* **528**, 207–211 (2015).
11. Levin, L. Laws of information conservation (nongrowth) and aspects of the foundation of probability theory. *Problemy Peredachi Informatsii* **10**, 30–35 (1974).
12. Solomonoff, R. J. A formal theory of inductive inference. part i. *Information and control* **7**, 1–22 (1964).
13. Dingle, K., Camargo, C. Q. & Louis, A. A. Input-output maps are strongly biased towards simple outputs. *Nature communications* **9**, 761 (2018).
14. Zenil, H., Badillo, L., Hernández-Orozco, S. & Hernández-Quiroz, F. Coding-theorem like behaviour and emergence of the universal distribution from resource-bounded algorithmic probability. *International Journal of Parallel, Emergent and Distributed Systems* **34**, 161–180 (2019).
15. Chibbaro, S., Rondoni, L. & Vulpiani, A. Reductionism, emergence and levels of reality. *Springer, Berlin. by SWISS FEDERAL INSTITUTE OF TECHNOLOGY ZURICH (ETH) on* **3**, 17 (2014).
16. Pérez, G. V., Louis, A. A. & Camargo, C. Q. Deep learning generalizes because the parameter-function map is biased towards simple functions. empharXiv preprint arXiv:1805.08522 (2018).
17. Mingard, C. *et al.* Neural networks are a priori biased towards Boolean functions with low entropy. *Arxiv preprint arXiv:1909.11522* (2019).
18. Rosenblatt, F. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review* **65**, 386 (1958).
19. LeCun, Y., Bengio, Y. & Hinton, G. Deep learning. *nature* **521**, 436 (2015).
20. Gács, P. *Lecture notes on descriptional complexity and randomness* (Boston University, Graduate School of Arts and Sciences, Computer Science Department, 1988).
21. Chaitin, G. Information-theoretic computation complexity. *IEEE Transactions on Information Theory* **20**, 10–15 (1974).

22. Calude, C. S., Salomaa, K. & Roblot, T. K. Finite state complexity. *Theoretical Computer Science* **412**, 5668–5677 (2011).
23. Grünwald, P. and Roos, T. Minimum description length revisited. *arXiv preprint arXiv:1908.08484* (2019).
24. Mathews, D. H. *et al*. Incorporating chemical modification constraints into a dynamic programming algorithm for prediction of rna secondary structure. *Proceedings of the National Academy of Sciences* **101**, 7287–7292 (2004).
25. Schuster, P., Fontana, W., Stadler, P. & Hofacker, I. From sequences to shapes and back: A case study in RNA secondary structures. *Proceedings: Biological Sciences* **255**, 279–284 (1994).
26. Hofacker, I. *et al*. Fast folding and comparison of RNA secondary structures. *Monatshefte für Chemie/Chemical Monthly* **125**, 167–188 (1994).
27. Fontana, W. Modelling evo-devo' with RNA. *BioEssays* **24**, 1164–1177 (2002).
28. Cowperthwaite, M., Economo, E., Harcombe, W., Miller, E. & Meyers, L. The ascent of the abundant: how mutational networks constrain evolution. *PLoS computational biology* **4**, e1000110 (2008).
29. Aguirre, J., Buldú, J. M., Stich, M. & Manrubia, S. C. Topological structure of the space of phenotypes: the case of rna neutral networks. *PloS ONE* **6**, e26324 (2011).
30. Schaper, S. & Louis, A. A. The arrival of the frequent: how bias in genotype-phenotype maps can steer populations to local optima. *PloS one* **9**, e86635 (2014).
31. Wagner, A. *Robustness and evolvability in living systems* (Princeton University Press Princeton, NJ:, 2005).
32. Wagner, A. *The Origins of Evolutionary Innovations: A Theory of Transformative Change in Living Systems* (Oxford University Press, 2011).
33. Dingle, K., Schaper, S. & Louis, A. A. The structure of the genotype-phenotype map strongly constrains the evolution of non-coding RNA. *Interface focus* **5**, 20150053 (2015).
34. Holcombe, M. and Holcombe, W. *Algebraic automata theory*, vol. 1 (Cambridge University Press, 2004).
35. Moore, E. F. Gedanken-experiments on sequential machines. *Automata studies* **34**, 129–153 (1956).
36. Almeida, A., Almeida, M., Alves, J., Moreira, N. & Reis, R. Fado and guitar: tools for automata manipulation and visualization. In *International Conference on Implementation and Application of Automata*, 65–74 (Springer, 2009).
37. Almeida, M., Moreira, N. & Reis, R. Enumeration and generation with a string automata representation. *Theoretical Computer Science* **387**, 93–102 (2007).
38. Rabin, M. O. & Scott, D. Finite automata and their decision problems. *IBM journal of research and development* **3**, 114–125 (1959).
39. Lempel, A. & Ziv, J. On the complexity of finite sequences. *Information Theory, IEEE Transactions on* **22**, 75–81 (1976).
40. Vilar, J., Kueh, H., Barkai, N. & Leibler, S. Mechanisms of noise-resistance in genetic oscillators. *Proceedings of the National Academy of Sciences of the United States of America* **99**, 5988 (2002).

Acknowledgements

K.D. acknowledges partial financial support from the Kuwait Foundation for the Advancement of Sciences (KFAS) grant number P115-12SL-06. G.V.P. acknowledges financial support from EPSRC through grant EP/G03706X/1.

Author contributions

K.D. and A.A.L. conceived the project. K.D., A.A.L. and G.V.P. derived the theoretical results. K.D. performed the numerical calculations for RNA, and G.V.P. performed the numerical calculations for the finite state transducer and the perceptron. K.D., G.V.P. and A.A.L. wrote the paper.

Competing interests

The authors declare no competing interests.

Additional information

Supplementary information is available for this paper at https://doi.org/10.1038/s41598-020-61135-7.

Correspondence and requests for materials should be addressed to A.A.L.

Reprints and permissions information is available at www.nature.com/reprints.

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.