ARTICLES
https://doi.org/10.1038/s42256-018-0005-0
nature machine intelligence

Causal deconvolution by algorithmic generative models

Hector Zenil[1,2,3,4,5,6*], Narsis A. Kiani[1,3,4,6*], Allan A. Zea[1,4,7] and Jesper Tegnér[3,5,6*]

Complex behaviour emerges from interactions between objects produced by different generating mechanisms. Yet to decode their causal origin(s) from observations remains one of the most fundamental challenges in science. Here we introduce a universal, unsupervised and parameter-free model-oriented approach, based on the seminal concept and the first principles of algorithmic probability, to decompose an observation into its most likely algorithmic generative models. Our approach uses a perturbation-based causal calculus to infer model representations. We demonstrate its ability to deconvolve interacting mechanisms regardless of whether the resultant objects are bit strings, space–time evolution diagrams, images or networks. Although this is mostly a conceptual contribution and an algorithmic framework, we also provide numerical evidence evaluating the ability of our methods to extract models from data produced by discrete dynamical systems such as cellular automata and complex networks. We think that these separating techniques can contribute to tackling the challenge of causation, thus complementing statistically oriented approaches.

We have introduced a suite of algorithms based on mathematical notions acknowledged to fully characterize the concept of randomness[1] . These algorithms enable us to study the algorithmic information dynamics of evolving systems, and to devise methods of reducing the dimensions of data[2] grounded on the first mathematical principles of randomness. Deconvolution of data by generative sources can be viewed as the ultimate goal of some supervised and unsupervised machine learning algorithms. This is a hard problem. Unsurprisingly, these approaches often lose sight of their goal of causal decomposition and instead seek to identify shared features of data, construed as evidence of their possible common origin. For example, in signal processing, popular methods such as *k*-means[3] or *k*-medoids[4] define heuristics based on the minimization of distance among data points according to a specific metric. Other popular methods, such as support vector clustering[5] and some traditional machine learning techniques, draw on probability distributions, regression and correlation techniques to achieve linear separation and produce different groups. For instance, most deep neural networks use gradient-based learning techniques to statistically map elements to a differentiable landscape. Another type of separation method relies on graph-theoretic properties. Methods that separate graphs by indices such as edge betweenness or the frequency of over-representation of certain subgraphs (for example, see ref. [6]), or by more sophisticated criteria such as shared graph spectral features, fall into this category[7–10]. All these methods make the assumption that, because the objects they study share statistical, topological or algebraic features[11], such objects may be generated by the same means or from the same sources.

Causality in machine learning

Despite their wide use and popularity, current approaches to causation are still both fundamentally and pragmatically dependent on linear regression and correlation tests, ranging from Granger's causality[12] to transfer entropy[13] and Pearl's interventionist do-calculus[14]. Nevertheless, considerable progress has been made in these areas, in particular through the introduction of Pearl's interventionist do-calculus[14], even when they are limited by their high dependency on probability distributions.

Our approach contributes to the discussion of disruptive techniques for introducing symbolic computation and causation into machine learning so as to better deal with hierarchically structured data and inductive inference. To this end, we combine techniques from perturbation analysis as introduced by Pearl[14] and algorithmic probability[15].

For illustration purposes, let us consider the area of convolutional neural networks, one of the most promising approaches to image classification in machine learning, in which a set of primitive features is extracted from a distribution of images. In contrast, or in addition to convolutional neural networks, algorithmic causal deconvolution requires separating features by their most likely common generative mechanisms rather than by their most discriminative statistical features, with those combinations that have the greatest variance serving to distinguish an image from all others. Synthesizing generative computer programs rather than using traditional pattern recognition amounts to producing a truly algorithmic explanatory generative model based on a deeper understanding of a causal mechanism than is possible through (non)linear regression. By way of analogy, storing angles in a deep convolutional neural network layer from a distribution of images does not (necessarily) mean that these angles put together in some order are responsible for generating the object itself, just as the digits of the Fibonacci sequence do not build the Fibonacci sequence by simply arranging themselves in the right order (something that may be difficult to scale and generalize), but are constructed from

[1]Algorithmic Dynamics Lab, Unit of Computational Medicine, Center for Molecular Medicine, Karolinska Institutet, Stockholm, Sweden. [2]Oxford Immune Algorithmics, Oxford University Innovation, Oxford, UK. [3]Unit of Computational Medicine, Center for Molecular Medicine, Department of Medicine, Karolinska Institutet, Stockholm, Sweden. [4]Algorithmic Nature Group, LABORES for the Natural and Digital Sciences, Paris, France. [5]Living Systems Laboratory, Biological and Environmental Sciences and Engineering Division and Computer, Electrical and Mathematical Sciences and Engineering Division, King Abdullah University of Science and Technology, Jeddah, Saudi Arabia. [6]Science for Life Laboratory, Karolinska Institutet, Stockholm, Sweden. [7]Escuela de Matemática, Facultad de Ciencias, UCV, Caracas, Venezuela. *e-mail: hector.zenil@ki.se; narsis.kiani@ki.se; jesper.tegner@kaust.edu.sa

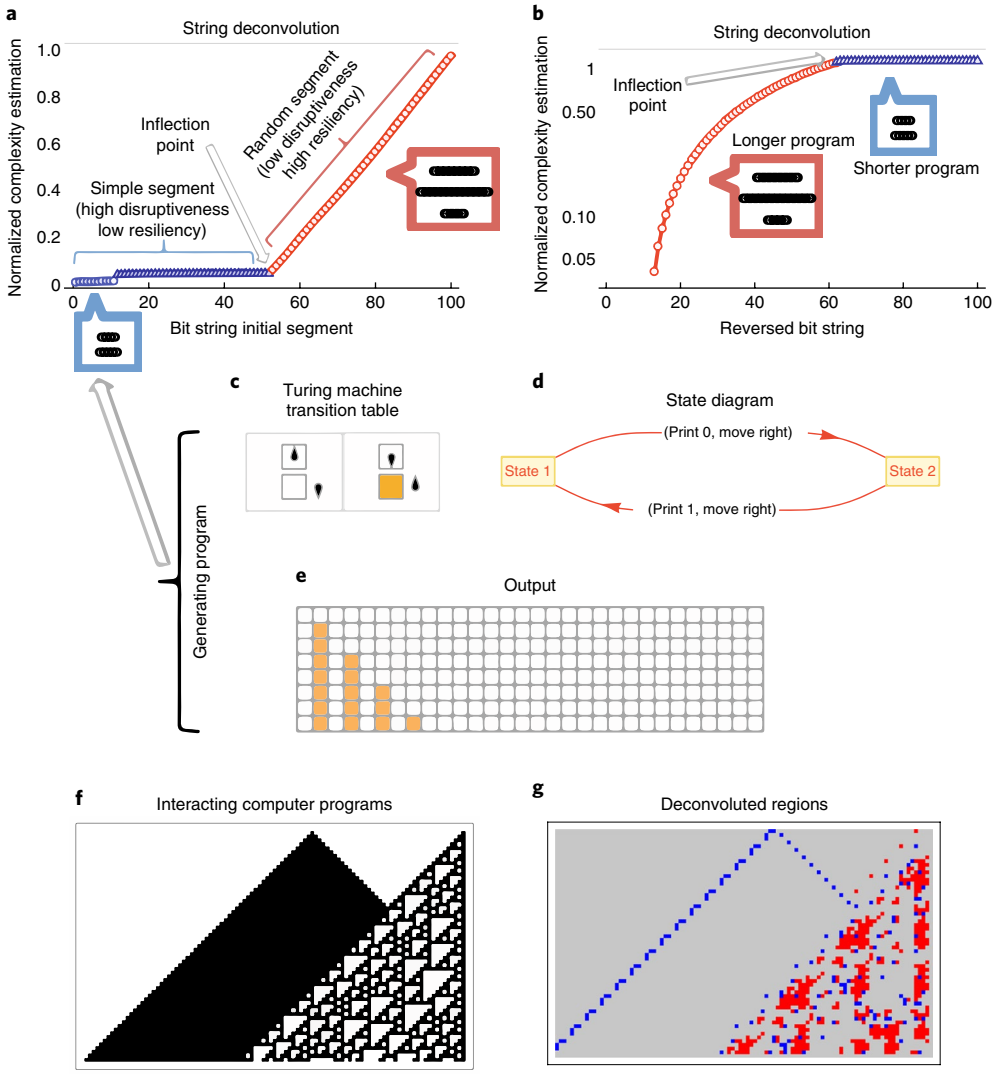

Fig. 1 | Proof of concept applied to a binary string composed of two segments with different underlying generating mechanisms (computer programs). **a**, Plot of complexity estimation of a regular segment (blue) consisting of the repetition of '01' 25 times followed by a random-looking segment (red). **b**, Log–linear plot reversing the order of **a**, yet preserving the qualitative behaviour of the different segments. We note that if the observation scale is at the level of single bits and they have a causal order according to individual observations, an order can be imposed over the generating programs to produce the string in the right order. **c**, The code of the smallest generating program (a non-terminating Turing machine) depicted visually (states are arrows in different directions), producing the string of 01^n for any n (0 is white and 1 is orange) starting from a blank tape, as shown in the space–time diagram (**e**). **d**, The same computer program as a state diagram. **f**, An illustration of a very simple case of interacting programs with one dominating the other, each with a different generating mechanism (ECA rules 255 v 110), each running for 60 steps using interacting rule 531 441 as described and explained in detail in the section 'Cellular automata' in the Supplementary Information. **g**, Algorithmic information footprint: every pixel is perturbed by flipping its value and evaluating its contribution to the original object coloured accordingly: grey represents no contribution, blue represents a low contribution and red a high contribution (its presence contributes to its algorithmic randomness).

an algorithmic process implementing a formula able to produce the digits of the Fibonacci sequence in the natural order according to a generative process.

We complement our work with techniques borrowed from classical information theory when we cannot do better, but at the core of our approach is the seminal concept of Solomonoff induction[15].

Information theory and complexity in machine learning

The notion of deconvolution is similar and related to the notion of decomposition in methods of pattern recognition[16], classical information theory[17] (a survey can be found in ref. [18]) and clustering by lossless compression[19] such as those methods based on information distances[20] and compression—for instance, the so-called 'normalized information distance'[21], and its related measure, the 'normalized compression distance'[22], and other variations.

Classical information theory[23] has provided techniques for capturing and encoding statistical properties from data that affect almost all areas of science. For example, mutual information captures various averages based on associated distributions of statistical properties that are contained in one variable about another; that is, it captures how information can be combined and decomposed in purely statistical terms. Recent developments based on information decomposition have been introduced[17,24] with the purpose of separating multivariate signals into their alleged generative sources. A proposal that has gained some traction is the so-called 'partial information decomposition'[17], which falls short[24], among other reasons,

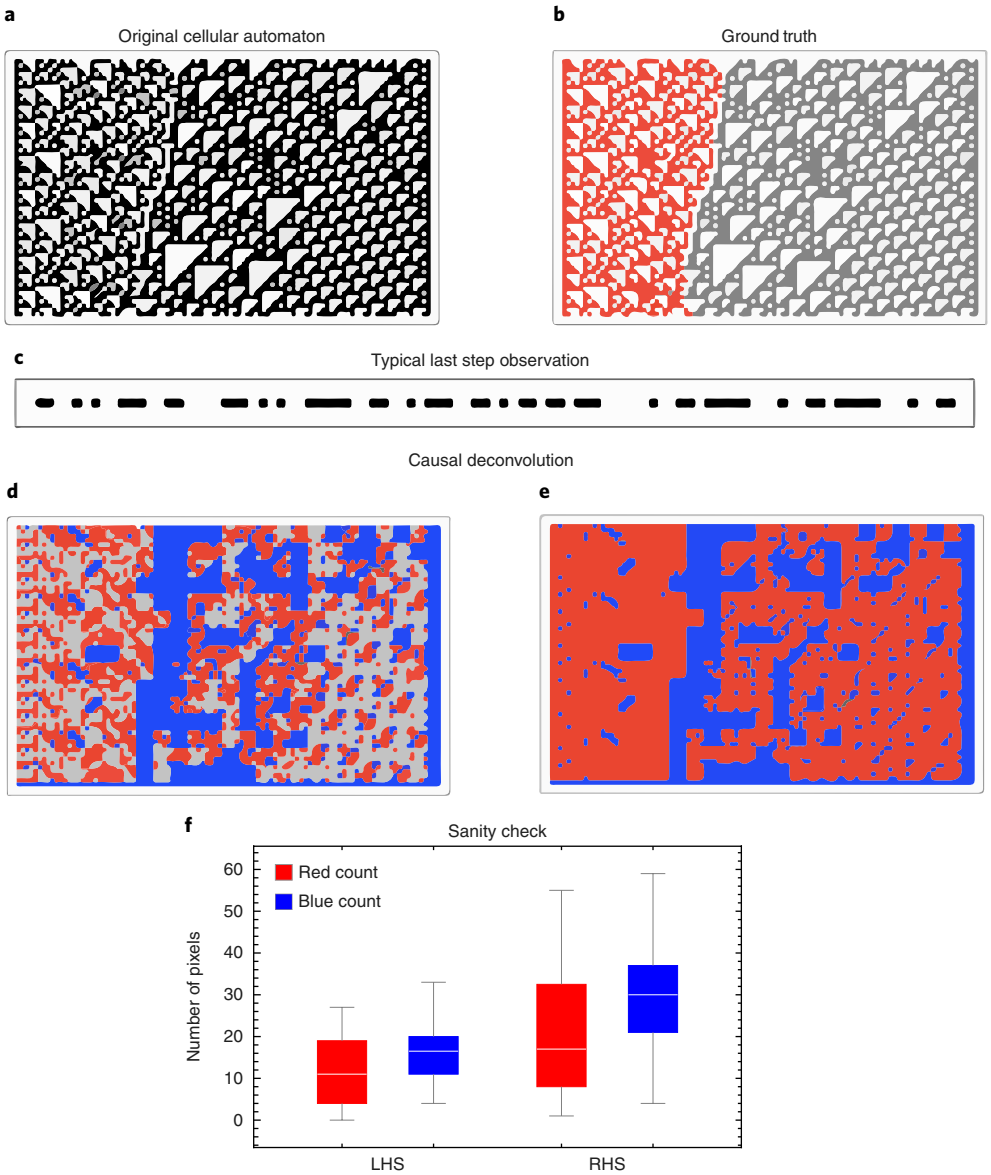

Fig. 2 | Training-free separation of intertwined programs despite their statistical similarity from an observer's perspective. a, The output of two different ECA (rules 60 and 110) with qualitatively complex output behaviour (11 to 60 steps depicted here, from a random initial condition) interacting with each other (one of which has been proved to be Turing-universal and the other of which has also been conjectured to be universal[1]), each producing structures of a similar type that, from an observer's perspective, are difficult to distinguish (see **c**). **b**, For comparison purposes, the two systems are distinguished in **b** (knowing the ground truth of which pixel is generated by which rule). **c**, What an observer of the last runtime would see in the form of a stream of bits with no clear statistical distinction. **d**, The algorithm pinpoints the regions of neutral, positive and negative, with the contiguous largest blue component segmenting the image into two. **e**, Only negative versus positive causal contributions occur where both Shannon entropy and popular lossless compression algorithms fail (see Supplementary Information). **f**, Sanity check/validation. Statistically significant quantitative differences between the parts divided by a contiguous line (shown in blue) between right and left occuring after application of the algorithm as illustrated in **e**, despite the weak statistical and qualitative differences between the two systems as depicted in **a**. The statistical difference shows that the method can help separate one system from the other. More cases are provided in the Supplementary Information.

because it can tell only what a variable can statistically reveal about some other variable.

Our approach can be viewed as replacing the methods used in computational mechanics[25,26] (traditionally based on, for example, Markov processes and Bayesian inference) with a measure based on algorithmic probability and an empirical estimation of the so-called universal distribution[27,28] (the distribution associated with algorithmic probability), while preserving the spirit of computational mechanics. In another related line of investigation into inductive inference, albeit mostly of a theoretical nature, there are approaches (such as AIXI) that combine algorithmic probability with decision theory, replacing the prior with the universal distribution[29], something that we have also proposed in the context of algorithmic cognition research[30].

In actual deployments, however, many approaches circumvent uncomputability and intractability by relying heavily on popular lossless compression algorithms such as Lempel–Ziv–Welch (LZW), minimum description length[31], Monte Carlo search and Markov processes, thereby effectively adopting weaker models of computation. Another related theoretical approach is Levin's

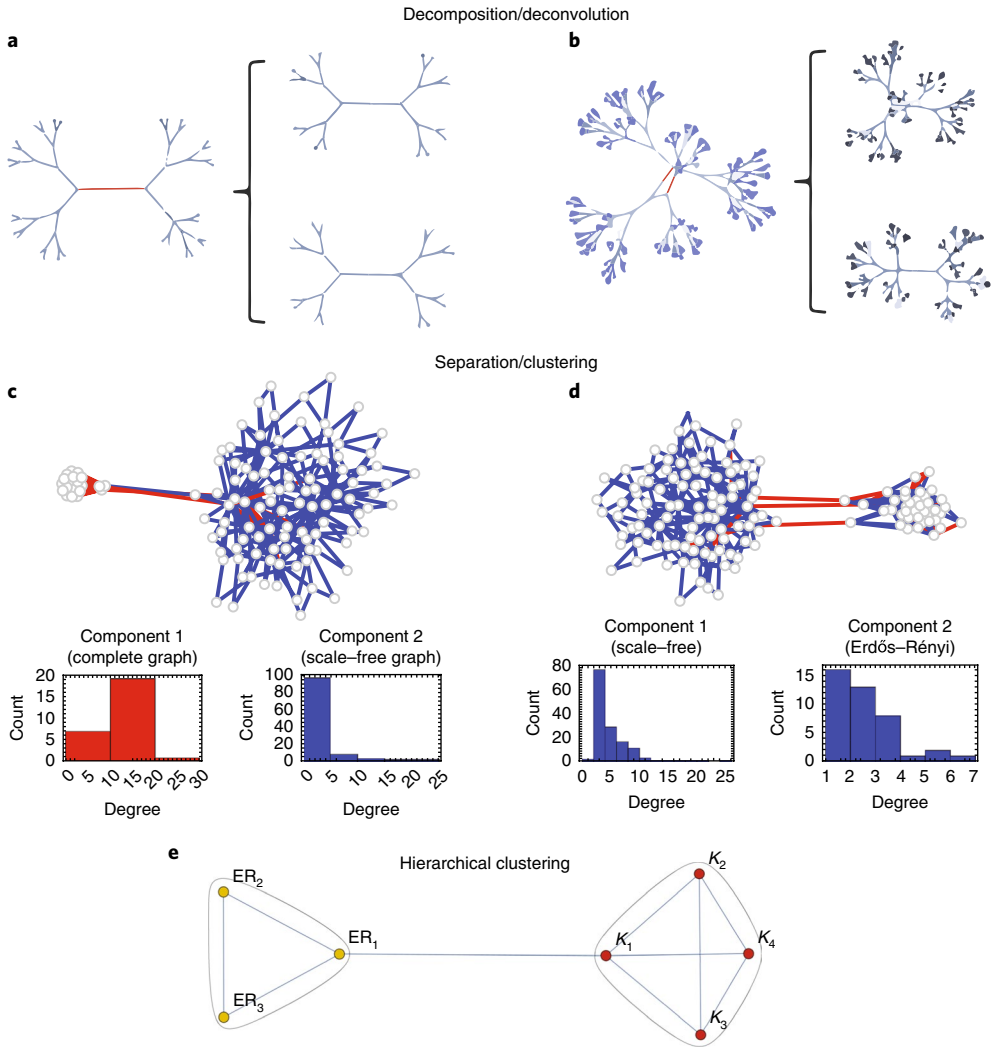

Fig. 3 | Algorithmic similarity and graph hierarchical decomposition leading to causal clustering. a,**b**, Forced deconvolution of a tree by minimization of graph algorithmic information loss, thereby maximizing the causal resemblance of the resultant components (hence causal clustering). Depicted are the components of *K*-ary trees of size 6 (a rooted tree in which each node has no more than *K* children) (**a**) and 10 (**b**) and their resulting graphs after one iteration of the deconvolution algorithm. **c**, Training-free deconvolution of 20 cases of scale-free networks generated by preferential attachment randomly connected to a complete graph. Negative edges break down the original graph into components corresponding to the different underlying generating mechanisms. Histograms correspond to each network according to the inferred decomposition showing the expected degree distribution (and how many times each degree occurs) of the two resulting major components after application of the method. **d**, Unsupervised deconvolution of 20 cases of random graphs (Erdős-Rényi) connected to scale-free networks (a typical case is depicted here). **e**, The algorithm first separates the subcomponents with the largest algorithmic difference, followed by other subcomponents, thus providing a natural hierarchy of source likelihood.

search[32] and variations[33], based on a dovetailing algorithm interleaving computer programs one step at a time, from shortest to longest, with each program assigned a fraction of time proportional to its probability during each iteration. Resource-bounded algorithmic (Kolmogorov–Chaitin) complexity[34] is also related to our approach. It imposes an upper bound on program length, which effectively adopts a linear finite automaton computational model[35], which we circumvent by allowing improvements while increasing the running time (though in practice each calculation is restricted to a resource-bounded calculation). In another category are methods of inductive inference such as computational measures of information gain and reinforcement[36,37].

Methods and algorithms

Algorithmic machine learning. One strength in our approach that Solomonoff himself defended[38] is that if machines are going to have a problem-solving capability similar to those of humans, machines should not start from scratch every time for each new problem. Therefore, we precompute and store estimations of the universal distribution from explorations of large sets of small computer programs that are able to explain small pieces of data. Each computer program represents a discrete generative model of its corresponding piece of data, which, when assembled in sequence, represents a full model. The precomputation allows practical applications to be implemented in linear time by querying a look-up table[27,28] acting as a working (Turing machine) tape and exchanging memory for computational time. This technique, combined with classical information theory in the way that the sequence model is assembled, provides key hints on the algorithmic content of a piece of data[39]. The estimations of our approach, denoted 'BDM', can always be improved on by running the original uncomputable process, with the advantage that partial results exponentially decrease the mul-

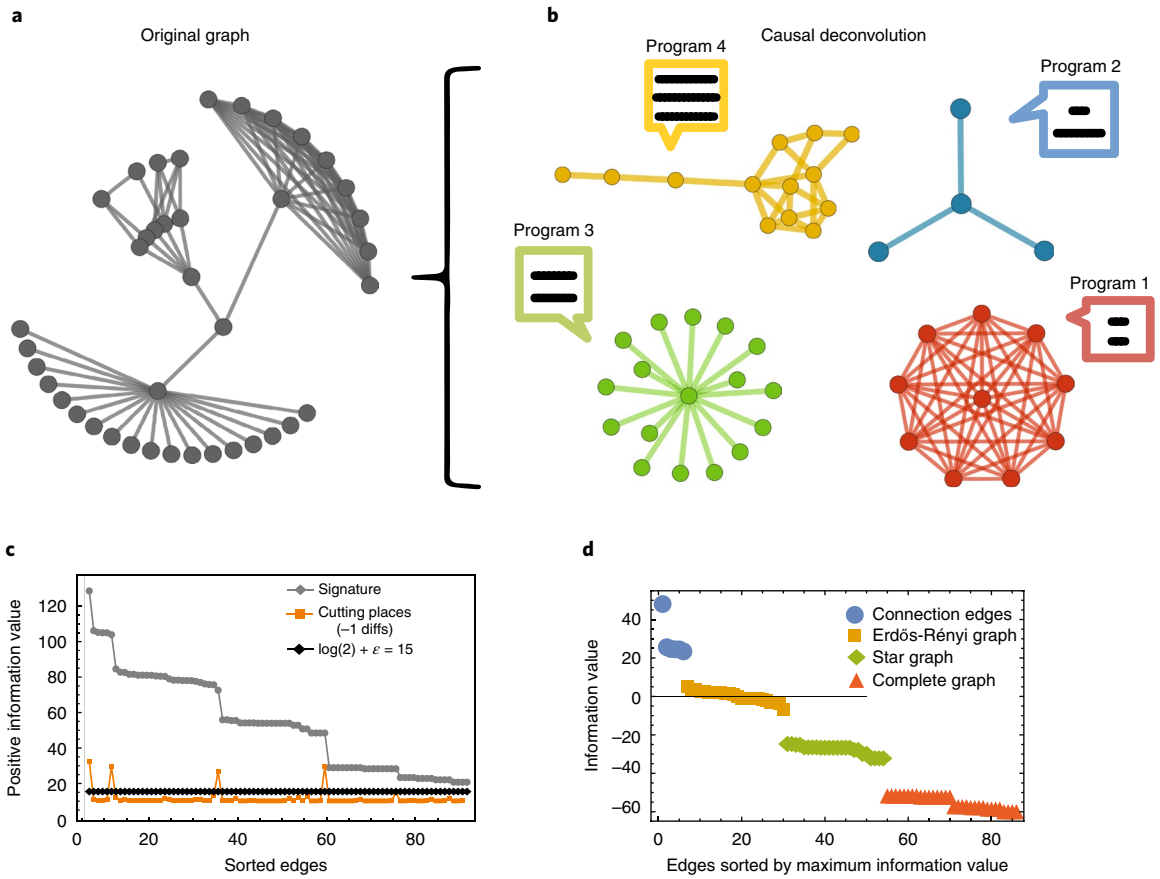

Fig. 4 | Unsupervised graph deconvolution identifies each different topological generating mechanism. a, Convoluted network composed of three subgraphs produced by different generating mechanisms. **b**, Causal deconvolution of **a** by algorithm (2), keeping edges connected if their removal does not produce a change in the algorithmic complexity of the original graph larger than $\log(2)+\varepsilon$. **c**, The 'information signature' (red line with circle markers) illustrates the distribution of information values for each edge (x axis) in the original graph (**a**). Also shown is a line of the differences of consecutive values of the signature multiplied by −1 (blue line with square markers), indicating the breaking points (the peaks that mark the edges to be deleted) with, in this case, four peaks (values) clearly standing out beyond the $\log(2)+\varepsilon$ line (orange rhombus) breaking the signature corresponding to the formation of each subgraph with high accuracy, thereby deconvolving the original graph (**a**) into the subgraphs (largest components) that are most likely to be generated by the same causal/algorithmic source. **d**, Signature decomposition according to the breaking points found in **c** imparting the colours to the subgraphs in **b** with results matching theoretical expectations.

tiplicative constant involved in approaches such as Levin's search. Our BDM approach (see Supplementary Information) has been shown to go beyond statistical capabilities[39], capturing features that cannot be grasped by methods such as Shannon entropy, traditional pattern recognition, or lossless compression.

Our approach to deconvolution, based on algorithmic information dynamics[1], is a bottom-up approach deeply rooted in Solomonoff's inductive inference[15,40,41]; it is motivated by previous approaches, but conceived and designed for scalability without compromising the power of the computational model too early on. Behind the number or sequence of numbers matching observations or data to a complexity value using our methods, we also offer access to the generating rules explaining the data as candidate models that can be used for validation against present and future data, thus enabling predictions.

At the core of algorithmic information dynamics[1] is the process of finding algorithmic candidate models in the form of computer programs able to explain pieces of observed data and group them by same/similar source or generating mechanism, using as a guide the length of the set of computer programs that produce each piece of data when a larger piece of data is decomposed in all possible ways by perturbation analysis. The main point is that if a computer program generates the data, different regions of the same data can be explained by the same algorithms.

Deconvolution algorithms. The algorithm in the context of networks is as follows. Let G be a graph and let $E=E(G)$ denote the set of edges. Let $G\backslash e$ denote the graph obtained after deleting an edge e from G. Let $C(G)$ be the estimation of the algorithmic complexity of G (see Supplementary Information). The 'information contribution' of e to G is given by $I(G,e):=C(G)-C(G\backslash e)$. A positive information contribution corresponds to information loss and a negative contribution to information gain. Here we wish to find the subset $F\subseteq E$ such that the removal of the edges in F disconnects G into N components and minimizes the loss of information among all subsets of edges, that is the subset such that $I(G,F)\leq I(G,S)$ for all $S\subseteq E$. Algorithm (1) in Supplementary Information allows us to obtain the subgraph $(V, E\backslash F)$ subject to the above conditions. The desired subset of edges is then given by $F=E(G)\backslash E(\mathrm{DECONVOLVE}(G,N))$.

The only parameter that the algorithm (see algorithm (1) in Supplementary Information) requires is N, the maximum number of components into which an object will be decomposed. However, there is a natural way to find the optimal terminating step, and therefore the number of maximum possible components that minimize

the sum of the lengths of the candidate generating mechanisms, making the algorithm truly parameter-free, as it is not required to have a preset number of desired components. We provide the pseudo-code (in algorithm (2) of the Supplementary Information) that determines the optimal number of components; it requires no N parameter.

Clearly, if components s_1 and s_2 have the same algorithmic complexity, this does not immediately imply that s_1 and s_2 are generated by exactly the same generating mechanism. However, because of the exponential decay of the algorithmic probability of an increasingly random object, we know that the less random an object is, the exponentially more likely it is that the underlying mechanism will be the same. This is because there are exponentially fewer short (far from randomness) programs than long ones (see Supplementary Fig. 9c). For example, in the extreme case of fully connected graphs, we see that the complete graph denoted by K_n, with n the node count, has the smallest possible algorithmic complexity growth as a function of n, that is, $\sim c+\log[n]$, where c is the length of the shortest computer program that implements the program that produces the matrix of size $n\times n$ with all 1 entries representing the complete graph. If $|C(s_1)-C(s_2)|\approx\log[n]$, and s_1 and s_2 are connected graphs, then s_1 and s_2 are with high probability produced by the same algorithm. Conversely, if $|C(s_1)-C(s_2)|$ departs from $\log(n)$, then the likelihood of being generated by the same algorithm exponentially vanishes. So the information regarding both the algorithmic complexity of the components and their relative size sheds light on the candidate generating mechanisms.

Algorithm unsupervised termination criterion. The algorithm suggests a natural terminating criterion. Let S be the object that has been produced by N mostly independent generative mechanisms. We decompose S into n parts $s_1, \ldots, s_n$ in such a way that each s_i, $i\in\{1 \ldots n\}$ has an underlying generating mechanism found by running the algorithm iteratively for increasing n. But after each iteration we calculate the minimum of the differences in algorithmic complexity among all components. The algorithm should then stop when the number of components is exactly N and the sum of the lengths—the estimated algorithmic complexity—of each of the programs diverges from the expected $\log[N]$), because the length of the individual causal mechanisms producing each new component will be breaking down a component that could previously be explained by the causal mechanism at an earlier iteration of the algorithm. An implementation of this idea for a graph is shown in algorithm (2) in Supplementary Information).

As a trivial example, let us take the string 1^n, where S^n means that the pattern S is repeated n times. After application of the algorithm, the terminating criterion will suggest that 1^n cannot be broken down into smaller segments, each with a different causal generating mechanism, the sum of whose total lengths will be shorter than the length of the generating mechanism producing 1^n itself. This is because the sum of the length of the shortest programs $\sum_i |p_i|$ running on a universal Turing machine generating segments of 1^n of length $m_i<n$ each, such that the concatenation $\cup_{i=1} p_i=1^n$ will be strictly greater than $C(1^n)$, given that each p_i halting criterion will require $i\log m_i$ bits more than $C(1^n)$.

Application to intertwined computer programs. A cellular automaton is a computer program that applies in parallel a global rule composed of local rules on a tape of cells with symbols (for example, binary). Thoroughly studied in ref. [1], elementary cellular automata (or ECA) are one-dimensional cellular automata that take into consideration in their local rules the cell next to the centre and the centre cell. For technical details see 'Cellular automata' section in the Supplementary Information.

Cellular automata offer an optimal testbed for our purposes because they are discrete dynamical systems able to illustrate their operation in a visual fashion. A cellular automaton can be interpreted as a one-dimensional object that produces a highly integrated two-dimensional image whose rows are causally connected and are thus ideal testing cases. Clearly, the deconvolution algorithms can be adapted to any object. In this case, instead of edge removal we apply row removal to the evolution of interacting cellular automata (see 'Cellular automata' section in the Supplementary Information for technical details). In what follows we perform experiments using interacting programs such as cellular automata as examples to illustrate the deconvolution algorithms. Interacting programs need to define how the interaction happens (see section 'Interacting CA' in Supplementary Information).

Numerical experiments

Behind our deconvolution methods is the idea that we can find a set of small computer rules or programs able to reconstruct a piece of data. Figure 1c,d illustrates this. In the deconvolution of a string generated by two different mechanisms (Fig. 1a,b) and thus in two different regimes (random versus non-random), computer programs such as those in Fig. 1c,d help to deconvolute the string. We then do the same in all other cases, but extending the number of degrees of freedom of a Turing machine tape. We note that our methods, as illustrated in Fig. 1a,b, are invariant to production or representation direction, given that the algorithmic probability and its reversal (and the entire set of computable transformations) are the same up to a small constant (the length of the computable transformation)[42].

Decomposition of sequences and space–time diagrams. We used different programs to produce different parts of a string, that is, a program p to generate segment s_1 and a program p' to generate segment s_2 put next to each other. Clearly, the string has been generated by two generating mechanisms (p and p'). Next we used the algorithm to deconvolve the string and find the number of generating mechanisms and most likely model mechanisms (the programs themselves), inducing a form of 'algorithmic partition' based on the likelihood of each segment being produced by different generating mechanisms.

Figure 1a–e illustrates how strings that have short generating mechanisms are significantly and consistently more sensitive to perturbations. The resulting string is **0101010101010101010101010101010101010101010101010101010101**1101001010101000000010011001111001100000111001100, with the bolding corresponding to the parts suggested by the different regimes, according to their algorithmic contribution and the segment's resilience in the face of perturbations (by deletion and replacement) of the original string. Behind every real number providing an estimation of the algorithmic complexity of a string or object, our methods also provide a set of generating computer programs able to produce such object when using our algorithmic-probability-based measure BDM.

Not only did we find the correct number of mechanisms (Fig. 1a,b), but also we found the candidate programs (which for this trivial example are exactly the original) that generate each segment (Fig. 1c–e) by seeking the shortest computer programs in a bottom-up approach[27,28]. Finding the shortest programs is, however, secondary, because we only care about the explanatory powers that different programs have to explain the data in full or in part, pinpointing the different causal nature of the segments and helping in the deconvolution of the original observation.

Figure 1c–e depicts the computer program (a non-terminating Turing machine) that is found when calculating the BDM of the 01^n string. The BDM approximation to the algorithmic complexity of any 01^n string is thus the number of small computer programs that are found capable of generating the same string or, conversely (via the algorithmic coding theorem, see refs. [39,43]), the length of the shortest program producing the string. For example, the string

01^n was trivially found to be generated by a large number of small computer programs (Fig. 1c,d depicts a non-terminating Turing machine with E its output) using our algorithmic methods (as opposed to, for example, using lossless compression, which would only obfuscate the possible generating model) with only two rules out of 2×2 rules for the size of Turing machine with only two states and two symbols and no more, and thus of very low algorithmic complexity compared to, for example, generating a random-looking string that would require a more complex (longer) computer program. The computer program of a truly random string will grow in proportion to the length of the random string, but for a low-complexity string such as 01^n, repeated any number of times n, the length of the computer program is of (almost) fixed size, growing only by $\log(n)$ if the computer program is required to stop after n iterations. In this case, 01^n is a trivial example with a strong statistical regularity whose low complexity could be captured by applying Shannon entropy alone on blocks of size 2.

Cellular automaton two-dimensional deconvolution. Figure 1f–g illustrates how the algorithm can separate regions produced by generating mechanisms of different algorithmic information content by observing their space–time dynamics, thereby contributing to the deconvolution of regions that are produced by different generating mechanisms. In this example, both programs are sufficiently robust to not break down (see section 'Robustness and limitations' in Supplementary Information) when they interact with each other, with rule 110 prevailing over 255. Yet, in the general case, it is not always easy to tell these mechanisms apart. In more sophisticated examples, such as in Fig. 2d,e, we see how the algorithm can break down contiguous regions separating an object into two major components, corresponding to the different generating computer programs that are intertwined and actively interacting with each other. The experiment was repeated 20 times using programs with differing qualitative (for example, Wolfram class) behaviour.

Figure 2f demonstrates that perturbations of regions in red have a more random effect after their application and are thus by themselves less algorithmically random. When regions are of the same algorithmic complexity they are likely to be generated by similar algorithms, that is, from algorithms that are of similar minimal length. The removal of pixels in the blue regions moves the interacting system away from randomness and the pixels are themselves more algorithmically random. Blue structures on the left-hand side correspond to large triangles occurring in ECA rule 110 that are usually used to compute and transfer information in the form of particles.

However, triangular patterns transfer information in a limited way because their light cone of influence reduces at the greatest possible speed of the automaton, and they are assigned an absolute neutral information value. Absolute neutral values are those closest to 0. Once separated, the two regions have clearly different algorithmic characteristics given by their causal perturbation sensitivity, with the right-hand side being more sensitive to both random and non-random perturbations. Moreover, Fig. 2f shows results compatible with the theoretical expectation and findings in ref. [1], where a measure of reprogrammability associated with the number and magnitude of elements that can move a dynamical system towards or away from randomness was introduced and shown to be related to fundamental properties of the attractor space of the system.

Figure 2c–f shows that by iterating the deconvolution algorithm not only do the two main components of the image correspond to the two generating ECA rules, but a second application of the algorithm would produce a third or more components corresponding to further resilient features generated by the rules, which can be considered rules themselves within a smaller rule (state/symbol) space. However, in the deconvolved observations, the interacting rule determining how two or more rules may interact effectively constitutes a third global rule to which the algorithm has no direct access, or an apparent region in the observed window.

In the case of Fig. 2, the terminating criterion retrieves $N=3$ components from the two interacting ECA (rule 60 and 110). This does not contradict the fact that we started from two generating mechanisms, because there are three clear regimes that are in fact likely to be reproducible by three different generating mechanisms, as suggested by the deconvolution algorithm itself, and as found in ref. [44], where it has been shown that rule 110 can be emulated by the composition of two simpler ECA rules (rules 51 and 118). As seen in Fig. 2, among the possible causal partitions, $N=2$ successfully deconvolves ECA rule 60 from rule 110 on the first run, with a greater difference than the difference found between $N=3$ components when breaking down rule 110 into its two different regimes.

Unsupervised network deconvolution. Clustering can usually be viewed as solving a problem which has an underlying tree structure according to some measure of interest. One way to think of optimal classification is to discover a tree structure at some level of depth, with tree leaves closer to each other when such objects have a common or similar causal mechanism and for which no feature of interest has been selected. Figure 3 illustrates how the algorithm may partition data, in this case starting from a trivial example that breaks down complete K-ary trees. Traditionally, both partitioning and clustering are induced by an arbitrary distance measure of interest that determines the connections in a tree, with elements closer to a cluster centre connected by edges. The algorithm breaks down the trees (see Fig. 3) into as many components as desired by iterating over remaining elements if required until the number of desired components is obtained or the terminating criterion is applied (see subsection 'Algorithm unsupervised termination criterion'). Figure 3a,b provides examples illustrating how to maximize topological symmetry. The algorithm can be applied, without loss of generalization, to any non-trivial graph, as in Fig. 3c,d, or to any dataset for that matter.

Figure 4 illustrates the algorithm and terminating criterion starting from an artificial graph composed of several graphs (two simple and one that is Erdős-Rényi random: a small Erdős-Rényi-random graph connected to a star graph and to a complete graph). The graph can be successfully decomposed by algorithmic probability (see Fig. 4d) by identifying the likelihood of an edge being produced by the same mechanism by virtue of being close to each other in the information contribution (which theoretically should be removed by only $\log(2)$ if it follows the normal evolution of the same process), hence what we call causal separation/partition and clustering. Figure 4d, with the broken components that were found above $\log(2)+\varepsilon$, also shows the distribution of edges coloured by graph membership, perfectly corresponding to the different subgraphs that were used to compose the original graph in Fig. 4a.

The same task using classical information theory (Shannon entropy) is shown not to be sensitive enough, and a popular lossless compression algorithm (compression based on LZW) provided a noisy approximation of the results obtained by using the block decomposition method, as defined in ref. [39] (see section 'Other methods and measures' in Supplementary Information for details).

Figure 3c–e illustrates how randomly connected graphs with different topologies can be broken down into their respective generative mechanisms. Figure 3c is a complete graph of size 20 randomly connected by three edges to a scale-free graph of size 100. The graphs are generated by different mechanisms. One is a small program that, given a number N of nodes, produces a graph with all nodes connected to all other $N-1$ nodes and has a program of short length that grows only by $\log N$[11]. The scale-free network is generated by the canonical preferential attachment algorithm with two edges per node and requires a slightly longer algorithm that grows by $\log N+c$ (ref. [11]), where c is a small constant accounting for the

pseudo-random choice of attachment nodes. The algorithm breaks the graphs into two components, each of which corresponds to the graphs with different degree distributions (depicted below each case) associated with its generating mechanism. This is because $|P(G_1)| + |P(G_2)| + \ldots + |P(G_n)| + |P(e_{G_i})| > |P(G_1G_2\ldots G_n)|$ for any G_i, where e_{G_i} is the set of edges randomly connecting G_i to G_j for any i and j for all G of low algorithmic complexity.

Figure 3d illustrates a case similar to that in Fig. 3c, but instead of a complete graph, an Erdős-Rényi graph with edge density 0.5 is produced and connected by three random edges to a scale-free network produced in the same fashion as in Fig. 3c. Again, the algorithm was able to break it down into the two corresponding subgraphs.

Conclusion

Current approaches to machine and deep learning are[45] ill-equipped to deal with inductive inference, explanation and causation. Our methods are different from those used in other approaches (even those based on lossless compression algorithms) to estimate algorithmic complexity, and in particular, those from classical information theory and other statistical traditions. The methods introduced here promote the use of techniques from causal and perturbation analysis[14] complemented by universal principles drawn from the theory of computability and algorithmic complexity.

Comparisons to other methods indicate that our approach is accurate and sensitive even in simplified form based on single-pixel perturbation (as opposed to, for example, full subset perturbations). This means there is also a lot of room for improvement and further exploration of other applications and other areas based on the same principles.

Our approach contributes to the discussion on ways to teach machine learning cause and effect, as recently called for by Pearl[45] so as to depart from traditional statistical approaches in machine learning. We strongly believe (just as did Minsky[46]) that moving in the algorithmic direction and introducing symbolic computation complemented by previous achievements in the area of causation such as counterfactuals and the perturbation analysis introduced by Pearl et al.[14], combined with the combinatorial power of current approaches to deep learning, is the way forward in addressing the challenge of causation in machine learning. Approaches like ours open up the possibility of designing more elaborate machine learning techniques with complementary and better equipped abilities grounded on completely different first principles.

Code availability. A basic online implementation is available at http://www.complexitycalculator.com/deconvolution. Implementations in R and the Wolfram Language are available at https://github.com/allgebrist/Causal-Deconvolution-of-Networks/ and https://www.algorithmicdynamics.net/software.html. Some functions are also included in the Supplementary Information (Supplementary Methods).

Data availability

The data that support the plots within this paper are available from the corresponding author upon request.

Received: 29 May 2018; Accepted: 5 November 2018;
Published online: 7 January 2019

References

1. Zenil, H. et al. An algorithmic information calculus for causal discovery and reprogramming systems. Preprint at https://doi.org/10.2139/ssrn.3193409 (2018).
2. Zenil, H., Kiani, N. A., Zea, A. A., Rueda-Toicen, A. & Tegnér, J. Data dimension reduction and network sparsification based on minimal algorithmic information loss. Preprint at https://arxiv.org/abs/1802.05843 (2018).
3. Lloyd, S. P. Least squares quantization in PCM. *IEEE Trans. Inform. Theory* **28**, 129–137 (1982).
4. Kaufman, L. & Rousseeuw, P. J. *in Statistical Data Analysis Based on the L1-Norm and Related Methods* (North-Holland, Amsterdam, 1987).
5. Ben-Hur, A., Horn, D., Siegelmann, H. & Vapnik, V. N. Support vector clustering. *J. Mach. Learn. Res.* **2**, 125–137 (2001).
6. Milo, R. et al. Network motifs: simple building blocks of complex networks. *Science* **298**, 824–827 (2002).
7. Newman, M. E. J. Finding community structure in networks using the eigenvectors of matrices. *Phys. Rev. E* **74**, 036104 (2006).
8. Benczur, A. & Karger, D. R. Approximating s-t minimum cuts in $O(n^2)$-time. In *Proc. Twenty-Eighth Annual ACM Symposium on the Theory of Computing* 47–55 (ACM, 1996).
9. Spielman, D. A. & Srivastava, N. Graph sparsification by effective resistances. In *Proc. Fortieth Annual ACM Symposium on Theory of Computing* 563–568 (ACM, 2008).
10. Spielman, D. A. & Teng, S.-H. Spectral sparsification of graphs. *SIAM J. Comput.* **40**, 981–1025 (2011).
11. Liu, M., Liu, B. & Wei, F. Graphs determined by their (signless) Laplacian spectra. *Electron. J. Linear Algebra* **22**, 112–124 (2011).
12. Granger, C. W. J. Investigating causal relations by econometric models and cross-spectral methods. *Econometrica* **37**, 424–438 (1969).
13. Schreiber, T. Measuring information transfer. *Phys. Rev. Lett.* **85**, 461–464 (2000).
14. Pearl, J. *Causality: Models, Reasoning and Inference* (Cambridge University Press, Cambridge, 2000).
15. Solomonoff, R. J. A formal theory of inductive inference: parts 1 and 2. *Inform. Control* **7**, 1–22–224–254 (1964).
16. Watanabe, S. in *Frontiers of Pattern Recognition* (ed. Watanabe, S.) 561–568 (Academic Press, New York, 1972).
17. Williams, P. L. & Beer, R. D. Nonnegative decomposition of multivariate information. Preprint at https://arxiv.org/abs/1004.2515 (2010).
18. Lizier, J. T., Bertschinger, N., Jost, J. & Wibral, M. Information decomposition of target effects from multi-source interactions: perspectives on previous, current and future work. *Entropy* **20**, 307 (2018).
19. Li, M. & Vitányi, P. M. B. *An Introduction to Kolmogorov Complexity and Its Applications* 3rd edn (Springer, New York, 2009).
20. Li, M., Chen, X., Li, X., Ma, B. & Vitányi, P. M. B. The similarity metric. *IEEE Trans. Inf. Theory* **50**, 3250–3264 (2004).
21. Bennett, C. H., Gács, P., Li, M., Vitányi, P. M. B. & Zurek, W. H. Information distance. *IEEE Trans. Inf. Theory* **44**, 1407–1423 (1998).
22. Cilibrasi, R. & Vitanyi, P. M. B. Clustering by compression. *IEEE Trans. Inf. Theory* **51**, 1523–1545 (2005).
23. Shannon, C. E. A mathematical theory of communication. *Bell Syst. Tech. J.* **27**, 379–423 (1948).
24. Ince, R. A. A. Measuring multivariate redundant information with pointwise common change in surprisal. *Entropy* **19**, 318 (2017).
25. Strelioff, C. C. & Crutchfield, J. P. Bayesian structural inference for hidden processes. *Phys. Rev. E* **89**, 042119 (2014).
26. Shalizi, C. R. & Crutchfield, J. P. Computational mechanics: pattern and prediction, structure and simplicity. *J. Stat. Phys.* **104**, 819–881 (2001).
27. Delahaye, J.-P. & Zenil, H. Numerical evaluation of the complexity of short strings: a glance into the innermost structure of algorithmic randomness. *Appl. Math. Comput.* **219**, 63–77 (2012).
28. Soler-Toscano, F., Zenil, H., Delahaye, J.-P. & Gauvrit, N. Calculating Kolmogorov complexity from the frequency output distributions of small Turing machines. *PLoS ONE* **9**, e96223 (2014).
29. Hutter, M. *Universal Artificial Intelligence* (EATCS Series, Springer, Berlin, 2005).
30. Gauvrit, N., Zenil, H. & Tegnér, J. in *Representation and Reality: Humans, Animals and Machines* (eds Dodig-Crnkovic, G. & Giovagnoli, R.) 117–139 (Springer, Berlin,Berlin, 2017).
31. Rissanen, J. Modeling by shortest data description. *Automatica* **14**, 465–658 (1978).
32. Levin, L. A. Universal search problems. *Probl. Inform. Transm.* **9**, 265–266 (1973).
33. Schmidhuber, J. The speed prior: a new simplicity measure yielding, near-optimal computable predictions. In *Proc. 15th annual conference on Computational Learning Theory (COLT 2002)* (eds Kivinen, J. & Sloan, R. H.) 216–228 (Springer, Sydney, 2002).
34. Daley, R. P. Minimal-program complexity of pseudo-recursive and pseudo-random sequences. *Math. Syst. Theory* **9**, 83–94 (1975).
35. 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. *Int. J. Parallel Emergent Distrib. Syst.* https://doi.org/10.1080/17445760.2018.1448932 (2018).
36. Hernández-Orallo, J. Computational measures of information gain and reinforcement in inference processes. *AI Commun.* **13**, 49–50 (2000).
37. Hernández-Orallo, J. Universal and cognitive notions of part. In *Proc. 4th Systems Science European Congress* 711–722 (EC, 1999).

38. Solomonoff, R. J. The time scale of artificial intelligence: reflections on social effects. *Human. Syst. Manag.* **5**, 149–153 (1985).
39. Zenil, H. et al. A decomposition method for global evaluation of Shannon entropy and local estimations of algorithmic complexity. *Entropy* **20**, 605 (2018).
40. Chaitin, G. J. On the length of programs for computing finite binary sequences. *J. ACM* **13**, 547–569 (1966).
41. Levin, L. A. Laws of information conservation (non-growth) and aspects of the foundation of probability theory. *Probl. Inf. Transm.* **10**, 206–210 (1974).
42. Zenil, H., Kiani, N. A. & Tegnér, J. Symmetry and correspondence of algorithmic complexity over geometric, spatial and topological representations. *Entropy* **20**, 534 (2018).
43. Zenil, H., Soler-Toscano, F., Delahaye, J.-P. & Gauvrit, N. Two-dimensional Kolmogorov complexity and validation of the coding theorem method by compressibility. *PeerJ Comput. Sci.* **1**, e23 (2013).
44. Riedel, J. & Zenil, H. Rule primality and compositional emergence of Turing-universality from elementary cellular automata. *J. Cell. Autom.* **13**, 479–497 (2018).
45. Pearl, J. To build truly intelligent machines, teach them cause and effect. *Quanta Magazine* (15 May 2018).
46. Minsky, M. The limits of understanding. *World Science Festival* https://www.worldsciencefestival.com/videos/the-limits-of-understanding/ (2014).

Acknowledgements

H.Z. was supported by Swedish Research Council (Vetenskapsrådet) grant number 2015-05299. J.T. was supported by the King Abdullah University of Science and Technology.

Author contributions

H.Z., N.A.K. and J.T. conceived and designed the algorithms. H.Z. designed the experiments and carried out the calculations and numerical experiments. A.A.Z. and H.Z. conceived the online tool to illustrate the method applied to simple examples based on this paper. All authors contributed to the writing of 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/s42256-018-0005-0.

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

Correspondence and requests for materials should be addressed to H.Z. or N.A.K. or J.T.

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