The problem of *Sophistication*

Peter Bloem, Steven de Rooij, Pieter Adriaans

image credit: Bec Brown, cloudsofcolour.com

This presentation is about the question *How do we quantify the amount of information in an object?* How do we formalize the intuition that some objects seem to contain more information than others, even if they have the same size?

How do we quantify information?

- Answer 1: Kolmogorov Complexity
- Answer 2: Sophistication

We have a very good answer to this question, in the form of *Kolmogorov complexity*. But, as we will see, Kolmogorov complexity doesn't always fit our intuition.

The second answer, *sophistication*, hopes to fix this. Sophistication is built on top of Kolmogorov complexity, goes by many different names, and as we will see, isn't nearly as well defined as Kolmogorov complexity. We've found some serious problems with Sophistication, and that's what this presentation is about.

overview

- Kolmogorov complexity
- Sophistication
- Problems for Sophistication
- Outlook for Sophistication

We will start with a brief introduction to Kolmogorov complexity, since sophistication is based on it. We will then have a look at sophistication itself: the basic idea and the different variants that exist. Then, we can get into the main issues we've discovered.

We will conclude with the outlook for sophistication: what conclusions can we draw? Is sophistication doomed, or is there some hope? And if some parts are broken beyond repair, can other aspects of the theory be salvaged?

Kolmogorov complexity

If I can fully describe an object in n bits, it contains at most n bits of information.

This is the intution behind Kolmogorov complexity in a single sentence. This leads very naturally to a measure of information content: take the shortest possible description of an object, the length of that description is the amount of information that the object contains.

Kolmogorov complexity

If I can fully describe an object in n bits, it contains at most n bits of information.

- object → bitstring
- describe → program on a universal computer

$$U(\bar{\imath}y) = T_i(y)$$

$$K^U(x) = \min\{|p| : U(p) = x\}$$

To formalize this notion, we need to be precise about what we mean by *an object* and by *a description*. For the object, we can simply assume that our objects are encoded into bitstrings in such a way that all the relevant information is captured. We can then build our theory as a measurement of the amount of information in bitstrings.

Secondly, we make no demands on the language used to describe these strings, save that it is *effective* and *Turing complete*. Or, equivalently, our descriptions are programs on some Universal Turing machine U.

properties

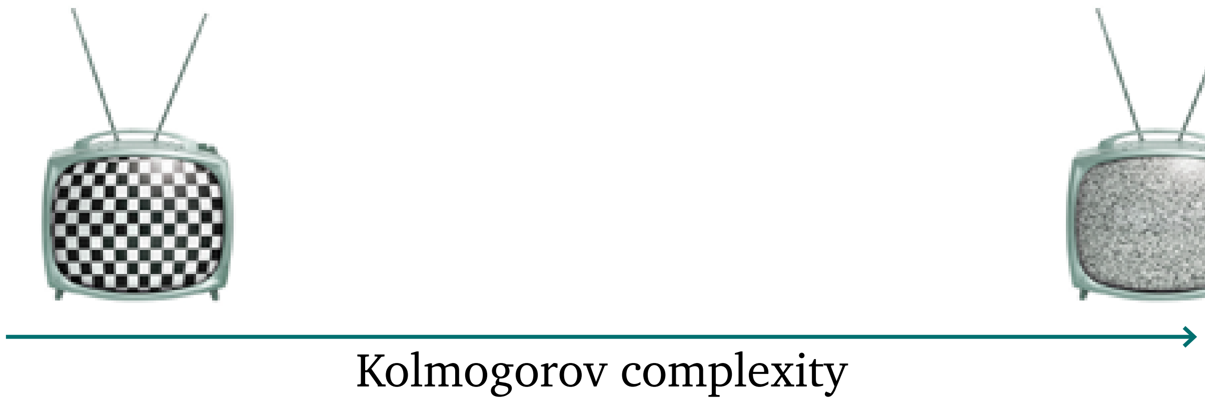
- K measures information
- K is unbounded
- K is **invariant**: $K^U(x) \overset{+}{=} K^V(x)$

There are several reasons why the idea of Kolmogorov complexity took off. In light of the comparison we are making with sophistication, the following are important. Firstly: it is very clear how the Kolmogorov complexity measures information. It reports a value in bits, and for each of those bits, we can tell exactly how the bit is used to encode the information in the object.

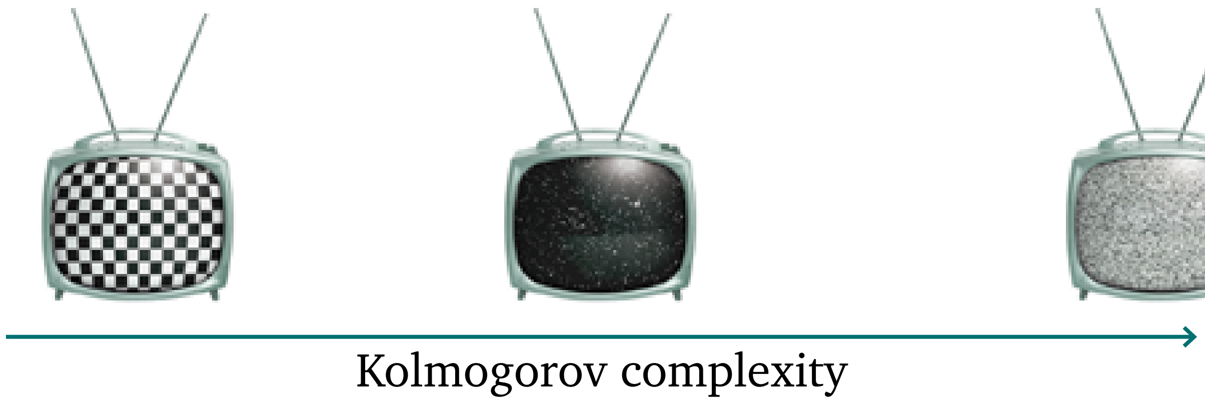
Secondly, the Kolmogorov complexity is unbounded. Intuitively, given some number n of bits, there is always some string containing more than n bits of information. Kolmogorov complexity does not violate this intuition.

Lastly, and most importantly, the Kolmogorov complexity is invariant. If we change the universal Turing machine used for our descriptions to another one, the value of the Kolmogorov complexity only changes in a limited and well-understood manner. To be precise, the value may change by any amount, but only by a constant independent of x.

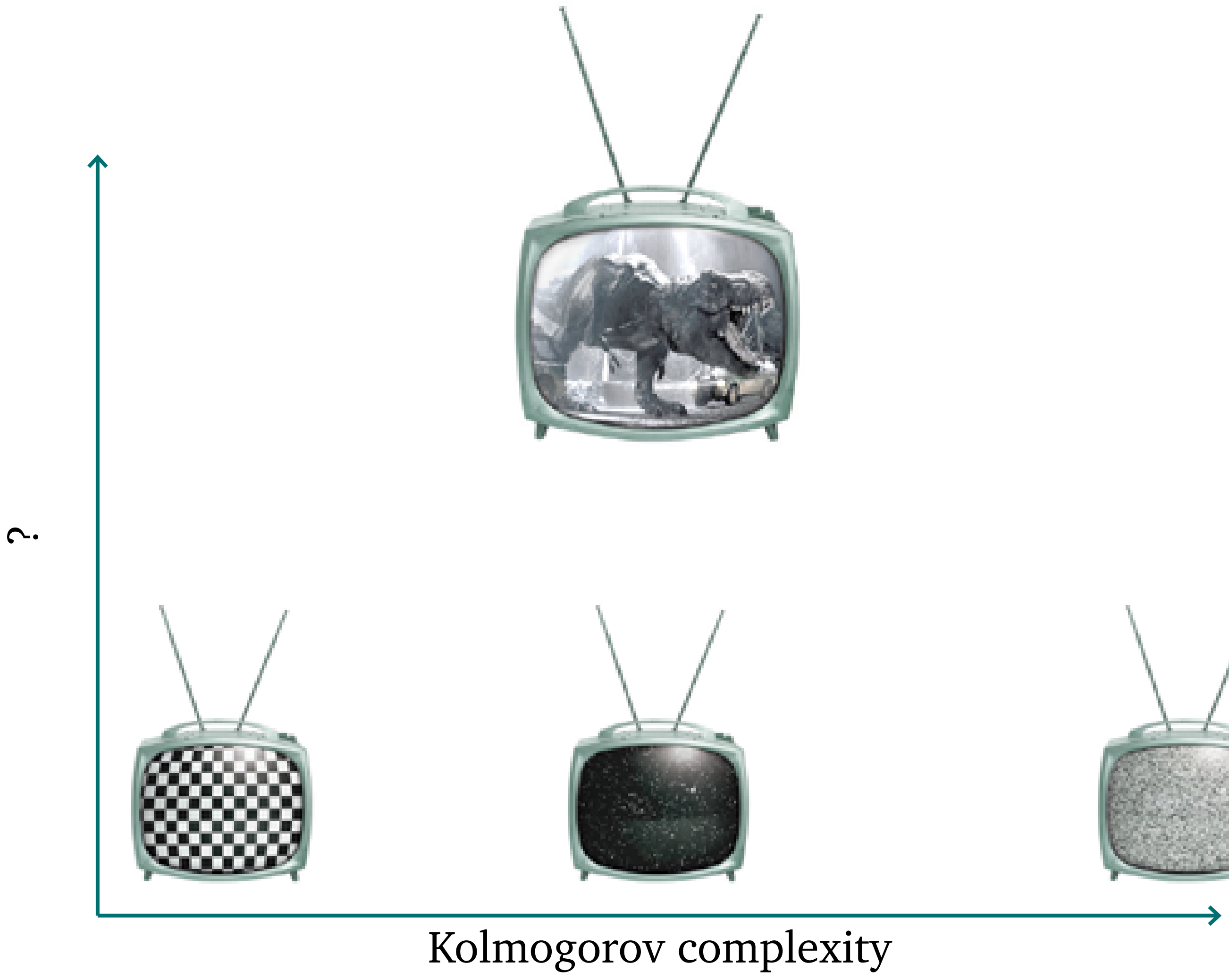
It is this invariance of Kolmogorov complexity that allows us to say that we are talking about *a property of the data*, and not just some arbitrary function computed on it. However we formalize the intuition behind Kolmogorov complexity, we always get the same answer,



So what kinds of things are complex and simple, by Kolmogorov complexity? Here we see two examples. On the left is a very *simple* television broadcast: a simple recurring pattern. The whole thing can be described very concisely. On the right we see the most *complex* possible broadcast: white noise. In this case, the only way to describe the broadcast is to provide for every pixel at every moment whether it's black or white.



To create something of medium complexity, we can take the noise and change the proportion of black pixels, to make the noise ‘darker’. Using basic compression techniques, we can use this imbalance to describe this signal more concisely than the white noise.



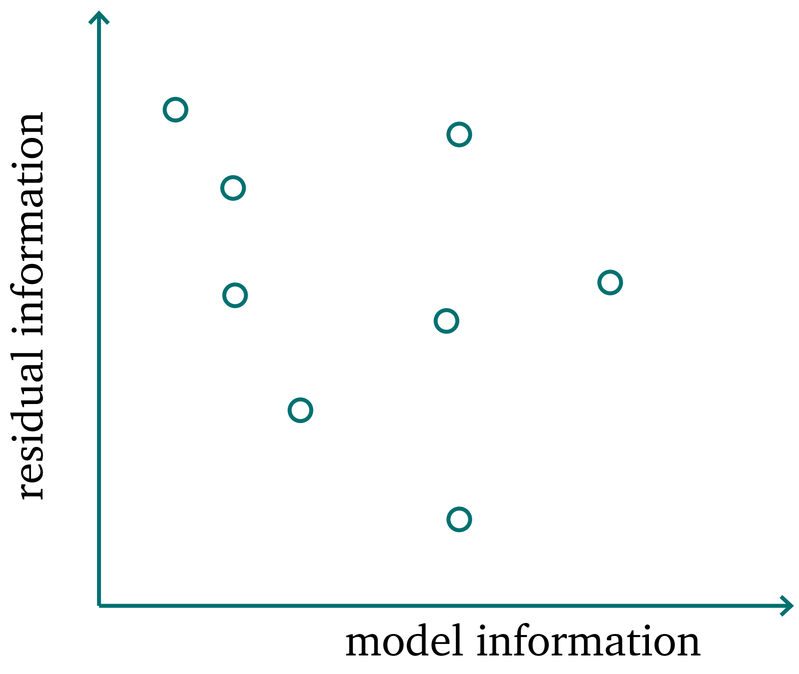
But none of these signals seem very rich to us. Some may be difficult to describe, and contain a lot of information, but we're unlikely to watch any of them for an extended amount of time. The information that they contain, isn't very *interesting*.

Signals that we are interested in are somewhere between the two extremes: they are partly predictable and partly unpredictable. They contain landscapes, human faces, dialogue, plot twists.

So, is there some method, in the spirit of Kolmogorov complexity, that will allow us to capture this vertical dimension? This is the question that sophistication hopes to answer.

Sophistication

the amount of *structured* information in a string



$T_i(y) = x$

T_i: model

y: residual information

(i, y): description of x

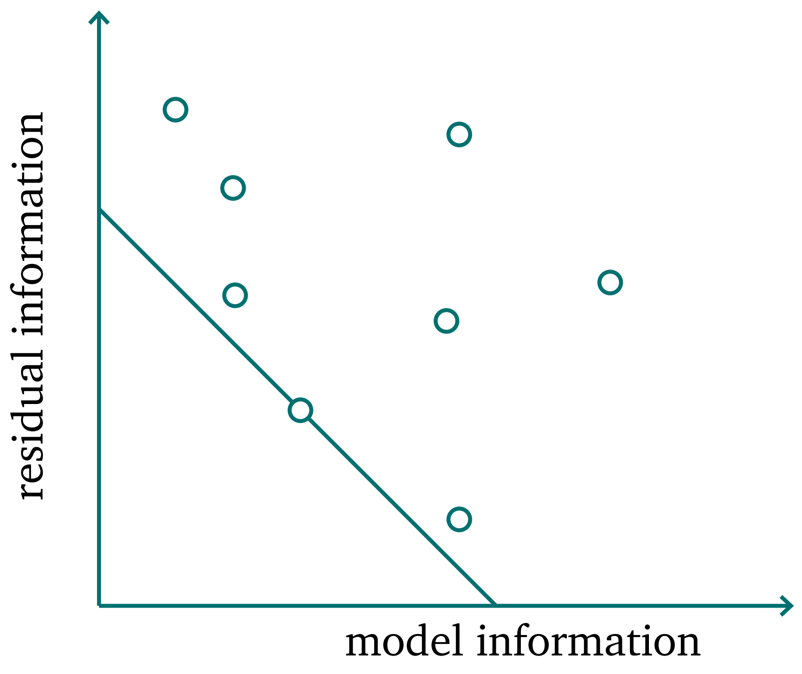
The basic idea of sophistication is not to measure all the information in a string but to split the information into a structural and a residual part. We do so by formulating a model class. We then describe the data by first describing the model, and then providing whatever information is needed to get from the model to the data. The sophistication, then, is the amount of information contained in the model: it counts only the structural information in the data. We call this *two-part coding*.

Which models are used differs between treatments of sophistication, but in all cases, we can think of the models as Turing machines, and of the residual information as inputs to the Turing machines.

Each dataset can be represented with many different two-part codings. We can visualize these with a scatter plot.

Sophistication

the amount of *structured* information in a string



$T_i(y) = x$

T_i: model

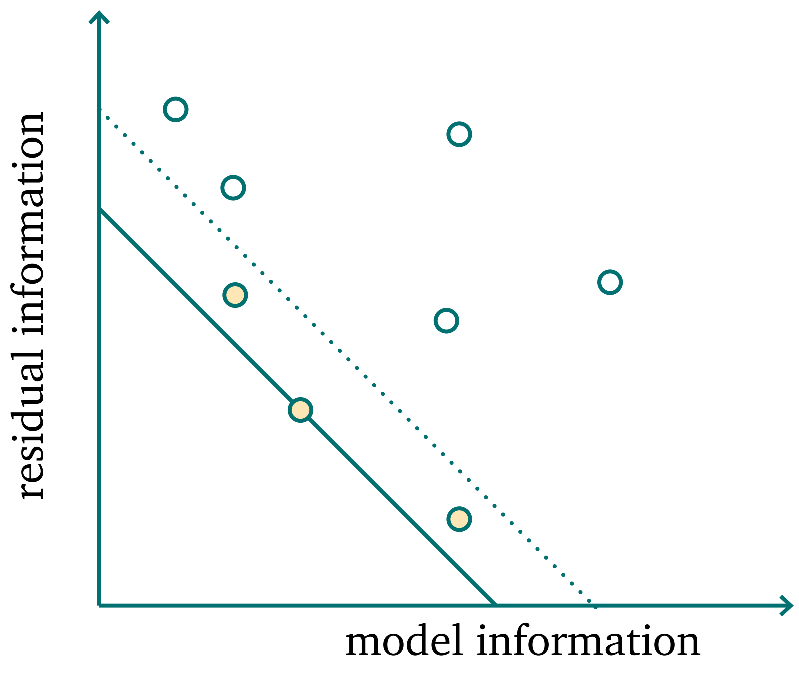
y: residual information

(i, y): description of x

By taking a 45° line, and sliding it up, we can find the most efficient two-part coding. If we allow all Turing machines, this two-part coding is the one that determines the Kolmogorov complexity.

Sophistication

the amount of *structured* information in a string



$T_i(y) = x$

T_i: model

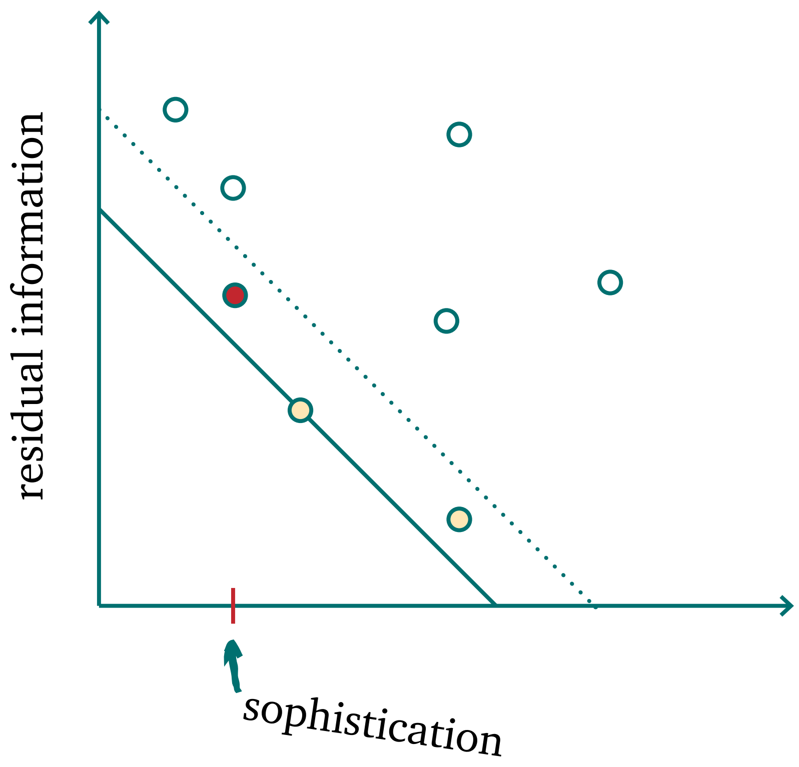
y: residual information

(i, y): description of x

We then allow a certain, constant slack. Any two-part coding within a given constant of the Kolmogorov complexity is taken into consideration. We call these the *candidates*.

Sophistication

the amount of *structured* information in a string



$T_i(y) = x$

T_i: model

y: residual information

(i, y): description of x

Among the candidates, we choose the representation with the smallest model. The amount if information in the model part of this representation is the sophistication.

Sophistication

the structure function

sophistication

facticity

(strong) algorithmic sufficient statistic

meaningful information

effective complexity

This principle has been proposed many times, by many different people, under many different names. We use sophistication as an umbrella term.

Among these people we find Kolmogorov himself, the authors of the standard textbook on Kolmogorov complexity, and a nobel laureate. Clearly, this is a strong intuition, at which many very intelligent people have arrived independently.

Nevertheless, we do not believe that this intuition is correct. We have found serious problems with all currently published proposals.

desiderata

- S(x) should measure *structural* information
- S(x) should not be bounded
    - K(x) - S(x) should also not be bounded
- S^U(x) should be **invariant** to the choice of U

In order to explain these problems, let’s return to the properties that make Kolmogorov complexity such a strong concept. If a formulation of sophistication is to be taken seriously, it should have the same properties.

First, it should clearly measure the structural information in a string. Second, It should not be bounded: intuitively, there should be no limit to the amount of structural information we can capture in a single string. Additionally, the difference between the Kolmogorov complexity and the sophistication should also not be bounded. This would make sophistication and Kolmogorov complexity equal, since we ignore constant terms.

And finally, and again most importantly, the sophistication should be invariant. If a change in the ad-hoc choices made in its construction, like the choice of universal Turing machine, will cause a large change in the value of the sophistication, we cannot claim that we are measruing a meaningful property of the data. We are simply computing an arbitrary function.

index sophistication

$$K(x) = \min\{|\bar{\imath}y| : T_i(y) = x\}$$

So let's look at some ways we can define sophistication, and how they go wrong.

The first, and possibly the most obvious option, is to 'open up' the Kolmogorov complexity. Because the Kolmogorov complexity uses programs on a universal Turing machine as descriptions, we are already minimizing over two-part descriptions internally.

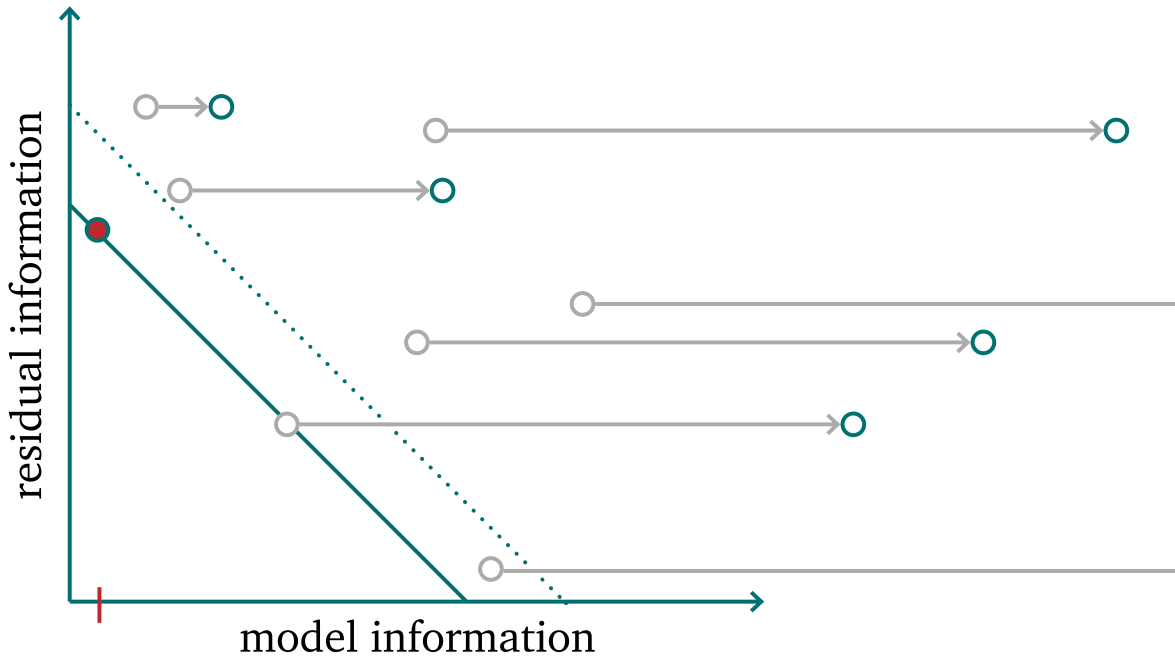
If we look at the shortest program for our data, the one whose length determines the Kolmogorov complexity, we see that its first bits encode a Turing machine, and the rest are the input to that Turing machine. This is simply how the universal Turing machine is defined.

So why don't we simply take the length of whits first part as the sophistication? There are several published proposals for sophistication that take this approach. Unfortunately, we can show that the consequences are disastrous.

the problem with index sophistication

$$V(0\ldots01y) = T_i(y)$$

2^i zeroes



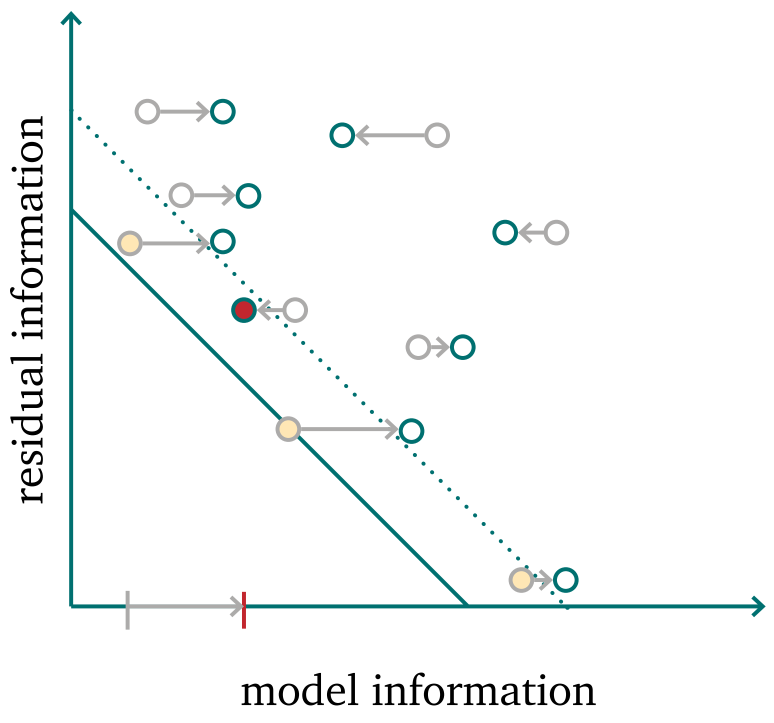
The problem is that the way a universal Turing machine encodes other Turing machines does not need to be efficient. To illustrate, let’s exaggerate the problem. Take some canonical enumeration of Turing machines, and define V so that it splits its input into a prefix consisting of a sequence of zeros followed by a one, and the rest of the string y. If the number of zeros is equal to 2^i for some integer i, the machine simulates Turing machine i with input y. Otherwise, it enters an infinite loop.

This is a perfectly valid universal Turing machine. We can use it to define Kolmogorov complexity, and the values we get, will be the same as we would with any other universal Turing machine, up to a constant. However, the models available will blow up exponentially. Even using a relatively simple Turing machine that could normally be described in 400 bits will require more storage than there is in the observable universe.

Why doesn’t the Kolmogorov complexity not suffer? Because it can use a more efficient universal Turing machine as its model. Unfortunately, this doesn’t help the sophistication. If we allow universal Turing machines as models, this choice will lead to a constant sophistication. If we somehow disallow universal Turing machines, the siophistication becomes highly dependent on how efficient our universal Turing machine is.

sophistication

$$K(f) = \min\{K(i) : T_i \text{ computes } f\}$$



$f(y) = x$

$K(f)$: model size

$|y|$: input size

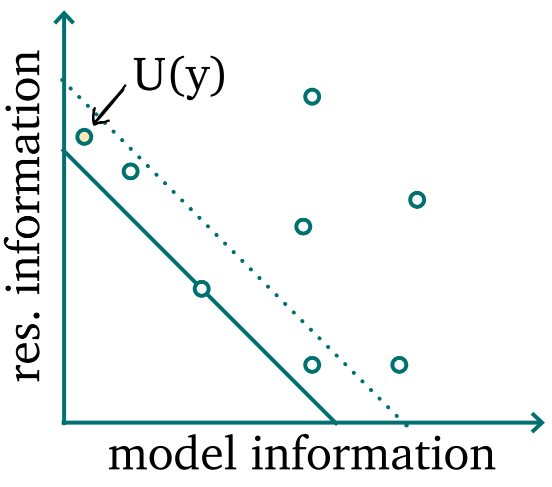
$K(f) + |y|$ total size

To solve this problem, we can translate the idea of Kolmogorov complexity to to models. Instead of counting the number of bits in the naive description of the model, we count the number of bits by which the model, or one equivalent to it, can be effectively described.

This “Kolmogorov complexity of models” is well-defined, and has the properties of slide 3, most importantly invariance. Thus, if we use K(f) to measure model information and build sophistication of this, we get a sophistication for which the model sizes only jump around by a constant under changes of the universal Turing machine.

Unfortunately, a constant jump in the model model size might still lead to much more than a constant jump in the sophistication, as illustrated here. Sicne the set of candidates is defined by a constant cut-off, a constant jump in model information may well push models in and out of the candidate set randomly, leading to arbitrarily large changes in the sophistication. Are such jumps possible in sophistication? We highlight two cases: underfitting and overfitting.

underfitting



- We can construct UTMs such that U is always in the candidate set
- solution: only total functions allowed

Underfitting is when too much information ends up in the residual part of the two-part code. We saw one extreme example already: if we use an universal Turing machine as a model, we get a representation that is guaranteed to be within a constant of the KOlmogorov complexity.

What is more, we show that there always exist choices of reference UTM such that this model is in the candidate set, resulting in a bounded sophistication.

This is not a new issue, and almost all treatments bypass it by explicitly disallowing universal models. The most common approach is to limit the model class to total functions.

underfitting

$U^A(p)$: simulate U(p) for at most *Ackermann*(p) steps.

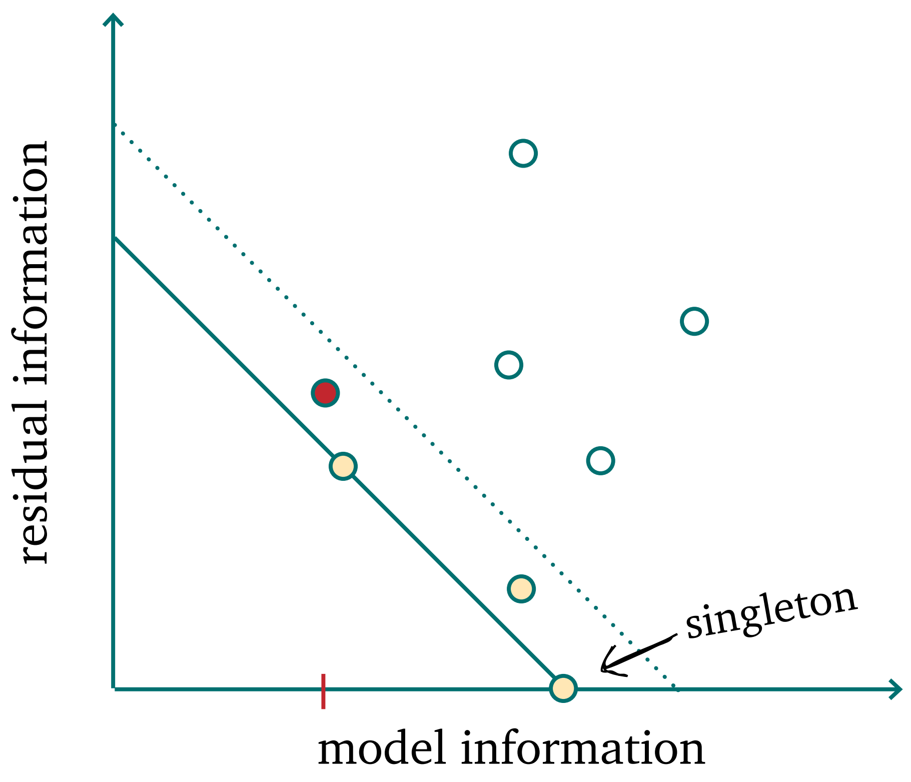
- Fixed size model.
- Reaches the Kolmogorov complexity for almost any “normal” string x.
- There are UTMs for which S(x) always selects U^A as a model (if x is normal)

Unfortunately, while this makes the problem more difficult to analyze, it is no solution. We can easily create a total version of U that behaves similarly for all but the most exotic strings. Here is an example: we give U a timebound of A(p). A grows very, very fast, so that for any program with a runtime that is not patently absurd, U^A behaves exactly the same way as U.

Thus, UA is always a potential model, and we can show that there are UTMs for which UA will always be selected as the model that degtermines the sophistication.

This means that for such sophistications, none of the structure that we find interesting, the dialogues, the landscape and the plot twists, will be counted towards the structural information. Only information that is so ‘deep’ that it would likely take longer than the lifetime of the universe to unpack will increase the sophistication.

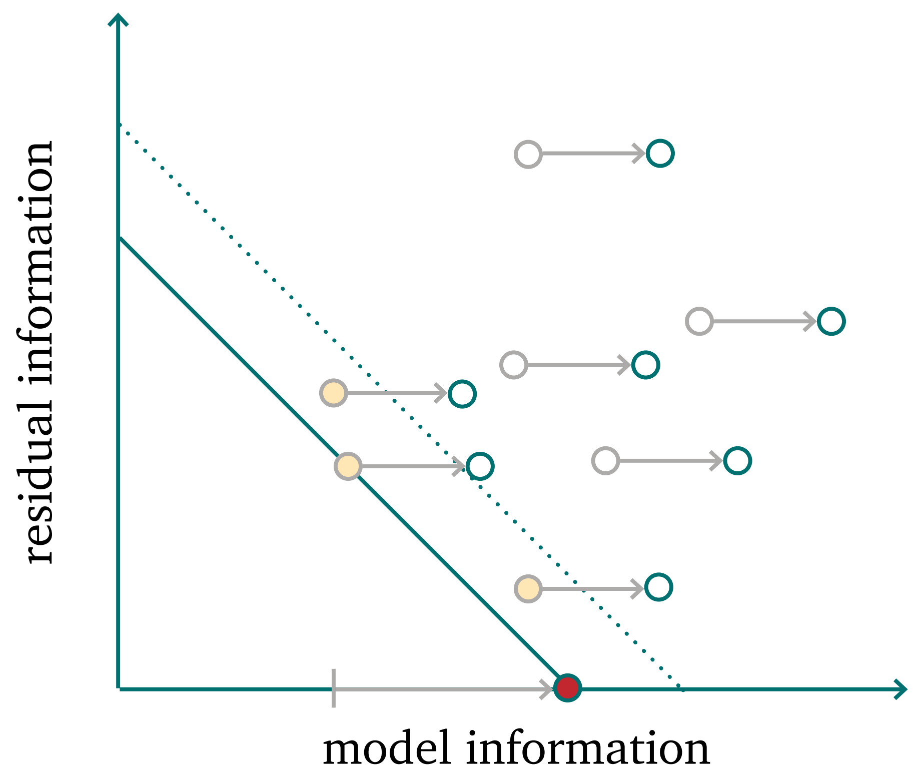
overfitting



On the other end of the spectrum we find *overfitting*. This is when too much information is counted as structure. In the most extreme case, the model encodes the entire data. Since we are using K(f) to measure the size of the model, the size of such a singleton model for x is equal (up to a constant) to K(x).

This has never bothered the authors of variants of sophistication so far, because the sophistication is always determing by the representation with the smallest model. The idea, presumably, is that there will always be representations with smaller models in the candidate set.

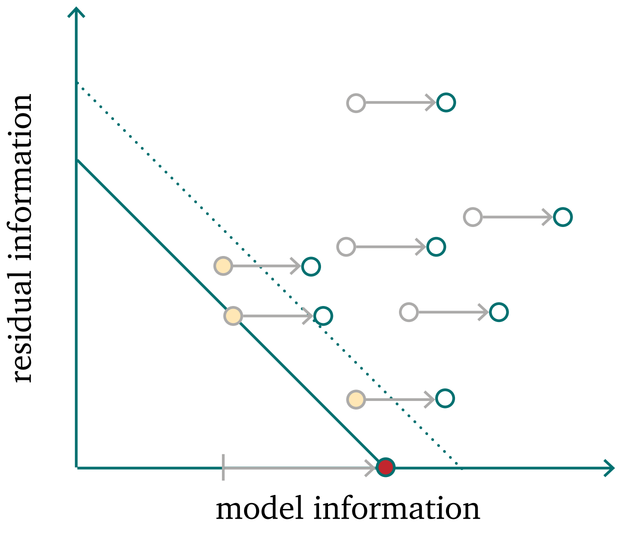
overfitting



Unfortunately, we can show that there exist UTMs for which this happens: all the models apart from the singletons get an arbitrary constant penalty. This means that all representations but the singletons get pushed out of the candidate set, and the singleton determings the sophistication.

This means that either the sophistication is always equal to the Kolmogorov complexity, or it is for some choices of UTM, and it isn't for others. Either way, one of the properties of slide 15 is violated.

overview



There exist UTMs for which the singleton models will always compress better than any other representation by an *arbitrary* constant amount.

recap

- inefficient indices
  - affects some definitions
  - disastrous, S(x) is highly non-invariant
- underfitting
  - affects *all* known variants
  - S(x) doesn't work as advertised
- overfitting
  - affects almost all variants
  - makes S(x) non-invariant

outlook

- Interesting results
  - absolutely nonstochastic objects
  - relation to depth
- A more thorough approach is required
- Can two-part coding really separate structural and incidental information unambiguously?

Where does this leave us? The investigation of sophistication has certainy been fruitful, even if the original aims have not quite been satisfied. However, if we wish to go forward with this idea, we must be more thorough in stating our definitions, desires, and proven properties.

Ultimately, the question boils down to separating structural and incidental information in an unambiguous manner. Our article provides several arguments for why we believe this to be a lost cause. However, these arguments are only informal, and we're happy to be proved wrong.

p@peterbloem.nl