Occam, Solomonoff, and neural networks as induction machines

The sun rose yesterday. It rose today. What gives us the right to say that it will rise again tomorrow?

Hume is famous for asking this question back in 1739. It has since come to be known generally as *Hume's Problem of Induction*. Two centuries later, the philosopher Nelson Goodman added a successor problem, which he christened the "new riddle of induction." It goes something like this:

Suppose all the emeralds you have seen so far appear green. You might jump to the conclusion that all emeralds are green. But wait a minute! What if we posed an alternative hypothesis, rather, that all emeralds are *grue* - meaning, quite seriously, that they are green before a time t somewhere in the future, and blue afterwards! Why can't we say that all emeralds are *grue* instead?

Though this example may initially seem contrived, Goodman is getting hinting at a very deep question about the nature of induction. For any set of data, multiple descriptions can fit the same observations. What this means is that a simple policy - "pick the one that the data supports" - is no longer helpful; the data supports multiple hypotheses, and each one to the same degree! What heuristic do we use to choose which one is better? Clearly, one is better (green is a more robust explanation than grue), yet where would such a heuristic come from?

This question will seem familiar to those of us who research machine learning. Many a model has fit the training set very closely - that is, each instance of the training data seems to "support" (i.e. not contradict) the model hypothesis - yet when tested on new data, fails. This is a phenomenon called overfitting, and it tends to come when your model is too complex; so complex that it manages to "memorize" the training set instead of finding the underlying pattern. Why does this approach fail?

For a given function (say, from n boolean variables to a boolean output), the number of possible inputs is exponential in n, and the number of unique functions that perfectly fit d datapoints scales with $2^{(2^n-d)}$. This quickly becomes intractable as n increases (referred to as the "curse of exponentiality"). And, if you attempt to generalize from boolean inputs to, say, integer inputs, there are an infinite number of "fitting functions" to choose from (in Nelson-speak, these might be: *grue at t, grue at t+1, grue at t+2 ... grellow at t, grellow at t+1 .... , gr-ellow-ue* (*green, then yellow, then blue, in that order*), gr-etc-etc). This is Goodman's true riddle of induction. How are we ever to choose between this infinite list of functions, when all of them are "supported" (in the generous sense of fitting the function) by the data? It is clear that some kind of heuristic is needed.

It's important to understand that it shouldn't matter whether this data is sourced from an image dataset, or from natural language, or from emerald observations. You have to choose a heuristic that is data-independent, i.e. a *universal prior.* Why? Well, since we defined the data as "everything with which the model fits", if we were to change our preferred hypothesis based on what type of data we used, it would already necessarily have been in the data! [1] This means that our heuristic of a good hypothesis cannot depend on any external information (like domain-specific knowledge, for example), or, alternatively, we assume this knowledge has already been encoded in the data somehow. This is the central riddle behind the process of induction; once we use *all of our data* to generate some hypothesis, we are still left with an exponentially large set of possibilities - to choose one requires some decision procedure that will not depend on the data.

A certain prescient 11th century friar, William of Ockham, might be of help today. He's famous for Ockham's razor, originally written as *Pluralitas non est ponenda sine necessitate*, or "Plurality is not to be posited without necessity." It's more commonly cited as "entities in explanations must not be multiplied beyond necessity."

I'd like to update this to make it a little more quantitatively robust. Here's my best translation into the language of today:

"For a given set of data, the best hypothesis to fit the set of data is the simplest one."

Or even more verbosely, "for a given set of data, and a given function that fits that data, we should weight our prior probability that the function was the ground-truth 'generator function' of the data as a function of its complexity."

So, first, how do we measure the complexity of a function? Here, we can get some useful tools from the field of algorithmic information theory - a fascinating sect of computer science that I very much wish Nelson Goodman had been exposed to. (he might have found the answers to his problems there!). The computer scientist Ray Solomonoff (who happened to be one of the "founding fathers" of AI, present in the Dartmouth conference of 1956), developed the notion of *Komolgorov Complexity* as a way to measure the "absolute complexity" of given data. The idea is that complexity [2] of any data set is a function of the length of the shortest program, in a pre-specified programming language, that produces that set of data as output. K-complexity is not strictly computable, yet is one of the most profound ideas in all of information theory. It explains why the bitstring

11111111111111111111111111111111111

is less complex than

10010110100101101001011010010110101O

For one, you can write a short program (i.e. [for i in range(20): print("1")]) to losslessly compress the data, and for other - a seemingly random string of bits - you are stuck with the humble [print("10010110100101101001011010010110101O")]. The former string is less complex than the latter - a fact that you intuitively feel - by virtue of the fact that it has *order*, patterns you can exploit to compress the data.

The power of this metric is immediately obvious. We now have a mathematically robust way to measure the complexity of the data generated by a given function. [3] Solomonoff was aware of the problem of induction, and harnessed k-complexity to introduce a formal version of Occam's razor, often referred to as *Solomonoff induction*. Its general statement goes something like this:

Among all "generating functions" that describe a set of data - in the sense that they begin by generating the given data - the *universal prior* that we should use to judge the likelihood of each one being the *true generating function* is inversely exponentially proportional to the k-complexity of the function. [4]

This is an information-theoretic expression of Ockham's razor. Solomonoff's answer to Goodman's "grue" question would be this: encode your dataset as a bit-string where each character represents whether or not the emerald you observed was green. For instance, here's the string so far - you have seen 22 emeralds in your life.

"GGGGGGGGGGGGGGGGGGGGGG"

To generate this string, you could try the "all emeralds are green" hypothesis: [5]

```
while true:
    print("G")
```

Or you could try the "all emeralds are grue" hypothesis:

```
for i in range(25):
    print("G")
while true:
    print("B")
```

And so on, for the infinite set of all functions, maximally compressed, that generate this string.

From here, we weight each hypothesis by how simple it is. In the case of Solomonoff induction, each additional bit in the function encoding multiplies the probability of that encoding by $\frac{1}{2}$ [6]. This is to ensure that the sum of all prior probability weights are guaranteed to converge, which is how we confront the enormity of the possibly infinite family of functions that can generate a given data set. Finally, we normalize to have the total probability of all viable generating functions sum to 1.

What we are left with is a set of prior probabilities that are independent of the data. This is the Solomonoff induction prior, and it represents the degree to which each hypothesis is a valid explanation of the given data. This is why functions that depend on some other arbitrary input (like time, in the 'grue' case) are worse; they are more complex! Every entity that you multiply beyond necessity adds to the complexity of the function, exponentially diminishing its likelihood under the Solomonoff metric.

This is also, in real-world terms, the reason why physicists rely on some degree of universality upon which to base their description of the world. They assume, for example, that the same laws of physics apply here as apply on mars, in deep space, and in your mother's living room. They might not realize it, but they are applying k-complexity bias by biological instinct. This is why they are skeptical when they hear reports of perpetual-motion machines; it adds an undue amount of complexity (in this case, physical laws have to "make an exception" for the claimant's machine, which means an extra set of parameters in the function that describes physical observations), implicitly reducing its likelihood. You might hear the physicist groan that it "doesn't make any sense" that the physical laws change near the claimant. What they mean is not that the new, perpetual-motion adapted set of physical laws does not describe the reported data (it may well do so!), but that making an exception for the area around the claimant's fiddly contraption breaks the simplicity of the "underlying assumptions" - namely, the shortest explanation that the laws of physics are invariant everywhere.

We are naturally drawn biologically to universality, to simplicity in inductive hypotheses. This is why we craft narratives, see patterns, and come up with explanations for events. [7] Evolution has learned over millions of years that simplicity is a good heuristic for prediction.

In fact, this is exactly the goal of a neural network. The fundamental problem of machine learning is not, as many would assume, to fit a set of data. To do this is trivial; just fit a function exactly to the data (i.e. overfit it), then do whatever you want with the predictions for inputs not covered in the training set. The core problem of machine learning is instead a *search problem* - the problem of approximating and choosing (from the often infinite space of possible functions), the simplest function that fits the data. In this sense, neural networks are, in a very fundamental way, automatic induction machines. They search a space for possible hypotheses (a very high-dimensional one), and use techniques to try and find the one that is most inductively plausible. For them (as for everyone), there exist only two criteria with which to judge the validity of a hypothesis: whether or not it fits the data, and how simple it is. There has been rich research about this topic; it is commonly referred to as the "bias-variance" tradeoff in ML theory.

To this end, researchers have developed a whole bag of tricks for making the functions they explore simpler. A common one is called the *regularization term* - a penalty added to the model's cost function that applies a simplicity bias to the search for a function fit. It penalizes complex functions, inhibiting the pernicious threat of overfitting from taking over. This is the theoretical intuition for why regularization results in better testing accuracy; simpler functions are more likely to approximate the true generating function. 10 This is why a well-designed neural network gains the ability to generalize to unseen data.

This is not to say that the simplest function is guaranteed to actually be the true function! Rather, that it is an example of a universal prior we can use to judge the likelihood of each in an intractably large set of functions. Yet merely stating the goal of the pursuit is much easier than actually developing an efficient solution. The theoretically best model generator would take every single piece of data it needs to train (taking into account even domain-specific knowledge), search through all possible programs that fit the data (an impossible task, not to mention imminently incomputable), and choose the one that has the highest Solomonoff prior. Sadly, we live in a finite world. The best we can do is to come up with good search algorithms: approximators for the ideal. [8] It is also worth mentioning that any prior that does not depend on the data will not account for noise; this is a vital point that throws any aim to compute naive Solomonoff priors (which rely on exactly fitting data) out the window. [9]

And there are still many questions that have gone unanswered. What, you ask, gives us the right to choose k-complexity (in a particular programming language, no less!) as our Solomonoff metric? [10] Why couldn't you choose a different one? And why k-complexity at all? Why not find some other convergent probability distribution?

Perhaps one would say something about how the k-complexity is the easiest, most obvious, most *simple* metric. And upon what heuristic do we say that simple metrics are best? Ah. I see Hume grinning at us from afar. The problem of induction still stands. Keep in mind that any coherent theory of induction - any heuristic that will solve Goodman's riddle - is itself a function, but this time a function of *functions*, mapping from each function to its Solomonoff validity metric. I'll work on it and get back to you, though as far as I know, Hume's original problem is yet unsolved.

But justification aside, this inductive method - Ockham's razor - seems to work pretty well so far! But why? Why does the universe seem biased towards the simplest generating functions for data? Is there some underlying "generator generator" that samples from the space of possible functions, where simpler ones are more likely? Do the laws of physics give us any hints?

I am not sure, but I do feel confident in saying that these might be among the deepest questions you can ask about the nature of truth. And in a way, there is something profound about the goal of machine learning as a field; to elucidate, in a truly epistemic fashion, the *best possible explanation* for a dataset. This process of information compression, of describing the data, of finding good (i.e. simple) explanations, is not mere statistical wizardry. Rather, it seems like learning in the truest sense. I guess they don't call it machine learning for nothing.

For more about learning (the nature of knowledge acquisition), read Knowledge as Information Compression, or Big Thing in the Chinese Room.

---

1. This is the rebuttal to those who protest this reductionist argument by saying that a model cannot always "know" how best to fit data; you need meta-information about the source of the data, assumptions about the data-collection process, etc. to make informed choices about which models to test and choose. While assuredly this is true when it comes to actually building and testing ML systems nowadays (and not likely to change in the near future), the theoretical argument assumes that the information you are basing your choices on has already been included in the data.

   An example of this might be the process of training models to approximate physical processes. Often, researchers will only choose those models that satisfy some basic physical principles (or adjust them, post-training) like conservation of energy, despite the training process allowing for models that lie slightly outside of these rigid laws. To the people that say that the simplest explanation for your data might not be the best, since it does not take into account domain-specific knowledge like the laws of physics; in this case, you *are* training it using the laws of physics; just doing it manually! If you wanted the hypothetical simplicity-optimizing machine learner to choose the best model, you would then have to also give it a plethora of information (like perhaps particle accelerator data, or results from physical experience) upon which we base our understanding of physics. In this case, the simplest explanation *would be* the one that included conservation of energy; it unifies all of the training data. [11] While impractical, this is the theoretical basis for induction. This is not to say that in practice you cannot customize your machine learning methods in different

ways depending on your data! It's another question altogether (and a fertile one) of how best to encode this information so that we can build genuinely practical machine learners. ↩

2. i.e. amount of information, i.e. entropy. ↩
3. This idea draws heavily on Turing's computability theory; the program also must halt. K-complexity is thus incomputable. ↩
4. There are a lot of parallels here to the idea of entropy; the most obvious one being that there are many more complex functions than simple ones. What we are searching for is, in this sense, the lowest-entropy generating function. ↩
5. (assuming naively that I've written the program with the smallest k-complexity in this pseudocode) ↩
6. Formally, we specify a Turing-complete Turing machine, then use the encoding of the function as the initial state of the memory tape to that Turing machine. We interpret what the Turing machine writes as the output generated by the function. The 1/2 comes from assuming that every new bit initialized makes the corresponding function half as likely. [12] ↩
7. Another essay coming about this topic: TL;DR: my initial guess is that the "language" of our minds has very concise "encodings" for ideas like purpose, cause, person, creator, etc, since these are very useful multi-purpose ideas to have in the jungle. This is why we (philosophers, especially) are so preoccupied with these ideas even though the laws of physics could not care less. ↩
8. This is why, in practice, certain methods (like ensemble models, for example), can have high k-complexity yet do very well on test sets. This is because they are only approximations, which may be close to the true generator despite being more complex. ↩
9. I plan to write a reflection about how, exactly, stochasticity might change the approach to finding a good hypothesis. I'm sure there has been literature on the topic that I have yet to read. ↩
10. In the general theoretic sense, what universal Turing machine are we using to simulate our generator? ↩
11. Indeed, the laws of physics are so well-entrenched by observation that we treat them as givens, though they are themselves empirical findings. This is why we would only consider the models that satisfy physical laws; we have a few centuries of data to back it up. If the model were to choose a function that violated our current understanding of physics, it would have to fit it also to that past data; if it did so in a more simple manner than our current laws, we might consider changing them! ↩
12. Yikes! I hear my "unfounded number" alarm going off! Why 1/2? We'd have to justify this by assuming some sort of probabilistic "generating function" uniquely sampled for arbitrary functions... perhaps we choose the simplest one? Resolving this circularity is left as an exercise to the reader. ↩