On entropy

tk waring

July 26, 2023

What is entropy?

Meanings are plural by nature — here's another one for entropy.

> Entropy is missing information.

This definition is essentially the beginning of the mathematical field of *information theory*, due to Claude Shannon in the 1940s. It's not, however, entirely separate from the version of entropy we see in thermodynamics. That is:

> Entropy is a measure of disorder in a probabilistic system.

What is probability? We say a flipped coin or rolled die is random, but that's not exactly true: if you knew exactly how quickly the coin was spinning, and exactly how long it would be in the air, then you could predict which side will come up. Of course we don't know that, so we think about probabilities, instead.

At another level of detail: we say a coin flip is *random*, because we can do the "same" experiment many times, and get different results. Of course, the experiments aren't the same (if the motion was exactly the same, we'd get the same result[1]), but as far as we know, they are. That is, probability is what's left when we take out some of the information.

In what way, then, does entropy relate to such missing information? Charmingly, some mathematicians measure information in units of "surprisals" — a message that contains more information is more surprising. Then, entropy measures how surprised you are by the outcome of your random experiment. (Strictly, how surprised you *expect* to be, since some outcomes might be more surprising than others.)

So far we haven't gone near "disorder". Here's how I conceptualise it. A probabilistic system is one about which we have incomplete information. That means, given what we know about it, there are many ways it could be arranged under the surface. For example, we might know the temperature and volume of a gas, but we don't know where each of the molecules is & how fast they are moving, &c &c.

[1] Don't bring up quantum mechanics, I'll cry.

Entropy measures this missing or supressed information. A more disordered system is one we know less about, one that has "more ways of seeming the same". If there's more information missing, we'll tend to be more surprised about the outcome.

Incidentally, this is why entropy always increases over time (the so-called *Second Law of Thermodynamics*) — in a choice between less entropy and more, there are *more ways* of doing the latter, so that's more likely. Scale this up to the size of the universe, and the first option becomes so much less likely as to be impossible.

Entropy and machine learning

I'm not sure I — or indeed anyone — really understands this connection, but given it was (sort of) the topic of my masters thesis I ought to give explaining it a go.

As you alluded to in your piece, the most effection "learning algorithms" we have use some randomness. In the transcript Miles talks about turning the little knobs, jiggling them around until the output gets better. There's a strong analogy between this process — jiggling the knobs around until the error gets smaller — and the way random physical systems move between the possible states, in this case tending to reduce the *energy* in the system (think: a ball won't roll up the hill).

The thing is, there are lots of ways for our thermodynamic system to have a small energy. The one that gets chosen is the one with the biggest entropy, simply because that's more likely, there are more ways of doing it.

In the context of, say, a deep neural network, a similar thing happens. There are *so* many knobs to spin, potentially more knobs (often called "weights" in this context) than there are examples to train them on. By rights, the network could easily pick weights that work perfectly for the training examples, but have no chance of working for anything else. In fact, conventional statistical wisdom says this is exactly what should happen.

However, when there are more weights than examples, the network is likely to end up with values for the knobs having a larger *entropy*. There is reason to believe that this is why they work at all. Even better, it seems that these higher entropy states give you simpler, smoother functions, and *that's why they work*. That, to me, is remarkable.

The elevator pitch for my thesis is that, in the way machine learning has been set up so far, we don't really have the language to figure out whether that claim is true. Like Miles mentioned we have no real idea what's going on "inside the head" of these "AI" algorithms, so how can we say if it's complex or simple. What I tried (am trying) to do is make a version of this process that comes out with an actual *program*, so that we can meaningfully talk about how complex it is, &c.

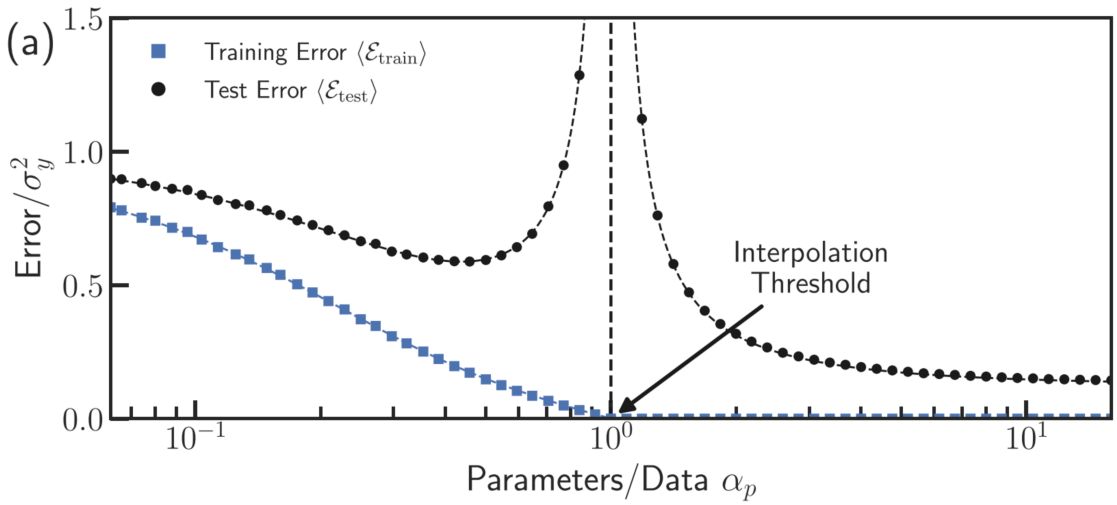

Figure 1: A plot from the wikipedia page on *double descent*. The U-shaped bit on the left is the conventional regime: once number of parameters (weights, knobs, whatever) gets close to the number of examples, it starts to overfit. That means the *training error* gets small, but it also learns stuff that isn't real, so the *test error* on examples it hasn't seen before is high. What's weird (and as far as I know unexplained) is that on the other side it starts getting better again.

Addendum: Turing

You mention the Turing test, and rightly say it's not a good test. On the other hand I can't resist a bit of Alan Turing apologia because I think he's incredible.

The history of AI infographic is good, and it mentions the paper where Turing introduces his test, in the 50s. It misses an earlier paper, from 1948, called Intelligent Machinery. Bear in mind this is at a time where computers had only just been made able to store their own programs (in 1946, also by Turing).

This paper anticipates that properly intelligent machines would have to learn for themselves, rather than being expert-taught. Not only this, he even suggests a kind of neural network as a way to do that. In 1948! Crazy.

He wrote this paper on a year-long sabbatical from Bell labs. The story goes he showed it to his bosses who dismissed it completely and so it was forgotten — until we come along decades later thinking we might have done something original only to find Turing got there first.