We Have Been Assimilated: Some Principles for Thinking About Algorithmic Systems

Paul N. Edwards

We have been assimilated: Some principles for thinking about algorithmic systems

Paul N. Edwards[1,2] [0000-0001-6228-1520]

[1] Center for International Security and Cooperation, Stanford University

[2] School of Information, University of Michigan

pedwards@stanford.edu

Abstract. This text is an opinion piece motivated by an invited keynote address at the 2018 IFIP 8.2 working conference, 'Living with Monsters?' (San Francisco, CA, 11 December 2018.) It outlines some principles for understanding algorithmic systems and considers their implications for the increasingly algorithm-driven infrastructures we currently inhabit. Four principles are advanced: the principle of (i) radical complexity, (ii) opacity, (iii) radical otherness, and (iv) infrastructuration or Borgian assimilation. It is hoped that these principles will help us develop a more critical appreciation of the emergent world marked by hybrid agency, accelerating feedback loops and ever-expanding infrastructures to which we have been all too willingly assimilated.

Keywords: principles of algorithmic systems; complexity; opacity; otherness; infrastruture

In 2016 I had the great good fortune to inherit a graduate seminar on "Algorithmic Culture" from my brilliant colleague Christian Sandvig, who had to back out of teaching it at the last minute. I already knew about half of the literature he assigned from other contexts, but I had never quite seen the pattern of it the way he did. That syllabus was full of intriguing surprises, one of those gifts from a random universe that happen a few times in your life if you're lucky.

The biggest surprise awaiting me, however, arrived in the classroom. Even sophisticated students with backgrounds in computer science could not, it seemed, genuinely grasp the complexity, opacity, partial autonomy, and radical otherness of algorithmic systems. Even fewer understood the stark differences between human-programmed algorithms and those produced by machine learning systems. Confronted with unsavory outcomes such as racial bias in face recognition systems, or pornographic images returned as top hits on the search "Black girls," their first, second, and third impulses were to attribute these results to the conscious or unconscious biases of human programmers, even to a deliberate intent to harm.

They are hardly alone. The most recent (and most ridiculous) expression of this impulse is President Trump's August 2018 claim that Google search results are biased

against him, accompanied by a thinly veiled threat of legal action against it and other tech firms. (Given his rampant narcissism, he might only be satisfied if 100 percent of all search results glorified his name.) Because people program computers, this line of thinking goes, the algorithms they create must reflect their biases, whether intentional or not.

This opinion piece has modest goals. First, I outline some principles for understanding algorithmic systems. I then consider their implications for the increasingly algorithm-driven infrastructures we currently inhabit.

The concept of an algorithm is often likened to a recipe or an assembly manual. Gather ingredients or parts, follow the instructions to the letter, and you end up with a Boston cream pie or an Ikea bookcase. In a computer, it's a list of instructions for operating on data. In computerese, algorithms are "effective procedures" that deterministically transform inputs into outputs, such that if you run the same algorithm on the same data it will always produce the same result — or at least that's how I learned it as a teenager in the 1970s, when I was programming and operating Honeywell mainframes for a large company in Maryland.

Lots of everyday algorithmic systems, such as your word processor or the accounting systems at banks, are still coded by human programmers and still correspond closely to the recipe view. If you dig into the code, you find thousands of sub-algorithms, each comprising a list of instructions for operating on data. Each one is entirely understandable in terms of human logic and/or mathematics. Apart from the occasional (and inevitable) bug, even the whole thing is pretty easy to grasp, because the interactions among all the sub-algorithms are part of the system design, and actually are not all that complicated. When I was learning computer science, the sub-algorithms were called "subroutines," for a good reason: they automate some step-by-step routine formerly carried out by a person. Make this word italic, put this text in a footnote, and so on.

Yet from a practical perspective, the recipe view of algorithmic systems is merely the kindergarten version of what's happening today. The largest modern climate models, for example, comprise over a million lines of code. Individual sub-models, constructed separately by domain specialists, represent the atmosphere, the oceans, sea ice, snow, land surfaces, and other elements of the climate system. Each model, in turn, contains numerous sub-models. An atmospheric model may include sub-models of (for example) radiation, cloud formation, aerosols, and atmospheric chemistry. Just as in the real climate system, all of these algorithms and sub-algorithms constantly interact, time-stepping forward in increments of 10 minutes until 100 simulated years have passed to generate a picture of how Earth's climate will evolve as we blast it with our greenhouse gases.

The complexity of climate models is reflected in complicated organizational structures required to create and maintain them. The Community Climate System Model, for example, holds annual meetings involving some 300 scientists and software developers, each representing larger groups responsible for different elements of the model [7]. Even though at least one specialist team understands each part, no one person understands the entire climate model in detail. Of course, this is the reason we need climate models in the first place: interactions in the real climate system are so many and so complicated that we can only understand them by simulating their behavior.

Let's call this inability to grasp all the interactions in a complicated algorithmic system the **principle of radical complexity**. This principle says that large, interactive algorithmic systems produce emergent behavior we cannot anticipate. Even if we can comprehend every individual component, scaling up highly interactive systems ultimately translates into cognitive opacity. It's still a recipe, and if you are a cook you can understand every instruction — but only the computer can bake the cake.

Arguably, an increasingly important category of algorithms — those produced by machine learning systems — no longer fits the recipe view at all. Machine learning comprises many techniques, but all share certain common features. "Learning" means building increasingly effective algorithms that can classify, or recognize, desired patterns in data. These patterns can be almost anything: handwritten letters, individual faces, fingerprints, email spam, consumer creditworthiness, and so on. The key difference is that these classification algorithms *aren't coded by human beings*.

Instead, human programmers write a "learner" algorithm, which generates other algorithms, known as models or classifiers. The learner algorithm — perhaps better understood as a "builder" — does not know in advance which models work best. (If it did, there would be no need for machine learning.) In some systems, the learner starts with a simple, approximate statistical model (aka classifier) introduced by the human programmer. It then constructs thousands of variations on the original classifier and tests all these variants against training data pre-sorted by the developers (e.g. spam/not spam, or handwritten letters identified as A, B, C, etc.). The best-performing classifiers are preserved, then modified to produce numerous new variants, which are tested again, and the cycle repeats. Once the classifier is trained, it's tested against "wild" datasets to prove that it can generalize beyond the training data. If it performs well enough, the cycle stops; if it doesn't, the training dataset may be expanded, and the test-modify-test-repeat cycle begins again. Thousands or millions of classifiers may be built and rejected before an acceptable level of performance is reached [21; 26].[1]

The nature of this process makes it all but impossible for human programmers to fully understand the logic that makes the winning algorithm work. "Learning" means finding significant features in the training data, which may include millions of examples. To discover these features, machine learning systems deploy n-dimensional matrices of *all* features (i.e. variables or properties) that might be relevant in the data. The classifier's goal is then to cross-correlate the features in all of these examples in search of some set or sets of features that reliably characterize the desired class. Many machine learning models use high-dimensional feature matrices — for example, the thousands of words that might indicate a message is spam — causing difficulties known as the "curse of dimensionality." These matrices, like the algorithms themselves, are beyond human comprehension [6].

Neural networks, another type of machine learning system, don't generate statistical models of feature correlation. Instead, they build algorithms "from scratch" by weighting the connections between simple artificial "neurons." When a given threshold of input values is reached, the neuron sends a signal to its neighbors. "Weights" or

[1] An excellent video explaining this process, ideal for classroom teaching, is available at https://www.youtube.com/watch?v=R9OHn5ZF4Uo.

multipliers (positive, negative, and/or fractional) on incoming connections determine whether each neuron fires; each outgoing connection is weighted as well. At every test of the network, threshold values and connection weights are automatically readjusted, and the system is tested again until it performs as desired. Such a system can comprise tens of thousands of neurons, arranged into multiple "layers," with millions of connections among them. Yet even a small network of a few hundred neurons can be capable of sophisticated pattern recognition. As in other machine learning techniques, a training data set includes human-classified examples of what the net is supposed to recognize; as it gets better, it more reliably distinguishes the positive examples from the negative ones.

Together, the weights, connections, thresholds, and layers in a neural net certainly comprise an algorithm — at least in the abstract sense of a list of instructions operating on data to produce a definite result. At the level of individual instructions, neural nets are incredibly simple, mostly reducing to simple addition and multiplication. Yet *this list of instructions is meaningless to a human being.* In other words, we don't really know exactly what they do, or how they do it. Following Jenna Burrell's lucid explication, let's call this the **principle of opacity** [4]. It's not only that machine learning algorithms are radically complex (though they are); it's that they cannot be understood as recipes at all.

In pursuit of some kind of grip on what neural nets are doing, some Google engineers realized in 2015 that they could reverse a neural network's pattern-recognition processes to make them *generate* images. Instead of asking a net to identify images of bananas (for example), they made it create its own image of a banana starting from white noise. The resulting images, often eerily beautiful and always strange, reveal just how differently a neural network "thinks." For example, one network generated images of dumbbells that always contained part of a human arm attached to the dumbbell (no doubt because most of its training images included people lifting weights).[2]

This kind of thing is commonplace in the world of machine learning. One neural net, trained to recognize sheep, did fairly well with this task most of the time, but also classified empty green fields as sheep. As one blogger put it, neural nets

> *…only see sheep where they expect to see them. They can find sheep easily in fields and mountainsides, but as soon as sheep start showing up in weird places, it becomes obvious how much the algorithms rely on guessing and probabilities. Bring sheep indoors, and they're labeled as cats. Pick up a sheep (or a goat) in your arms, and they're labeled as dogs. The thing is, neural networks match patterns. They see patches of furlike texture, a bunch of green, and conclude that there are sheep. If they see fur and kitchen shapes, they may conclude instead that there are cats. If life plays by the rules, image recognition works well. But as soon as people — or sheep — do something unexpected, the algorithms show their weaknesses.*[3]

[2] https://ai.googleblog.com/2015/06/inceptionism-going-deeper-into-neural.html

[3] http://aiweirdness.com/post/171451900302/do-neural-nets-dream-of-electric-sheep

A fascinating collection of bizarre logic from the closely related fields of evolutionary computation and artificial life includes the example of a program that was supposed to learn to sort lists. Instead, it deleted the list — which was then (from its point of view) no longer unsorted. Another program built simulated robots tasked with learning to travel quickly; one such robot assembled itself into a tall tower, then simply fell over to "travel" a long way. Another algorithm "was supposed to figure out how to apply a minimum force to a plane landing on an aircraft carrier. Instead, it discovered that if it applied a *huge* force, it would overflow the program's memory and would register instead as a very *small* force. The pilot would die but, hey, perfect score."[4] Machine-built systems use machine logic, not human logic. Let's call this the **principle of radical otherness**.

As in the cases above, sometimes error analysis can give us clues to how that logic may work. All too often, however, significant errors only become apparent after an algorithm created by machine learning is deployed operationally. One example is the well-known case, reported in 2015, of the Google Photos system labeling images of African-Americans as gorillas. Google apologized and said it would "fix" the algorithm. But in early 2018, nearly three years later, tesing by *Wired* magazine revealed that Google had simply blocked its image recognition system from labeling *any* photo as a gorilla or chimpanzee. (It will recognize other primates, but not those.)[5] The radical otherness of the recognition system's logic makes it impossible for engineers to tweak it directly, or to know exactly why it cannot distinguish between an African-American face and a gorilla — a difference that is immediately obvious to people. (Of course, we must admit that we don't know how people manage this feat either.) It's possible that Google is just being lazy and hasn't even tried to fix the algorithm. But another possible explanation is that even if a re-trained algorithm is much less likely to make this mistake, the social costs of repeating it even once are too great for the company to unblock the "gorilla" tag. This is just one of many examples of how machine learning can go awry.

The last principle I want to articulate here is hard to name succinctly. It stems from the fact that we now live in a world governed not by algorithmic systems *per se*, but rather by interacting ecologies of algorithmic systems, human individuals, social groups, cultures, and organizations. Natural ecosystems are characterized by extensive interactions and feedbacks among species. Changes in one species (caused by disease, parasites, etc.) or in the physical environment (higher temperatures, less water) affect all the others to varying degrees, driving some to extinction while creating openings for new species to enter the mix. In the long run, all species evolve in response to changing conditions, including the simultaneous evolution of other species.

Similarly, in today's information ecosystems, everything interacts with almost everything else, constantly evolving and adapting to changing conditions. People and organizations are always part of byzantine algorithmic feedback loops. Your "waste" data of searches and clicks becomes input to Google, Amazon, and Facebook, which feed it back to you as targeted advertising and personalized search. Bots spread both true and

[4] http://aiweirdness.com/post/172894792687/when-algorithms-surprise-us

[5] https://www.wired.com/story/when-it-comes-to-gorillas-google-photos-remains-blind/

fake news on Twitter, but humans retweet the fake news more often [2; 25]. New forms of "networked discrimination" — much more fine-grained than the traditional broad categories of race, class, and gender — become possible when individuals' online social networks can be viewed by prospective employers and used to determine "cultural fit" [3]. The essential materials of culture itself, such as music, movies, books, TV, games, and informal social interaction, are exchanged, "curated," recommended, and sometimes even produced by algorithmic systems, a phenomenon Galloway and Striphas have called "algorithmic culture" [10; 11; 15; 24].

Online recruiting makes it possible to hire employees, or enlist terrorists, without ever meeting them in person, challenging the traditional concept of an "organization." People develop folk theories of how algorithmic systems work, leading to quasi-superstitious practices that may in turn influence algorithmic behavior [5; 9; 20]. Algorithmic systems designed by the so-called "gaming" industry very effectively reinforce and "optimize" gambling addictions [22]. Meanwhile, YouTube, Facebook, Instagram, and hundreds of other so-called "platforms" promote other addictions, including the "outrage cycle" currently driving American politics. Some of this becomes embedded in culture as norms and expectations, reinforcing and legitimizing both addictive behavior and awful politics. I'm sure you could easily name a hundred other ways in which algorithmic systems, culture, individuals, social groups, and organizations interact.

This is true not only at the level of societies, but even within the larger platform systems. It's popular to talk about "the Google search algorithm," for example. That phrase makes many of us immediately think of PageRank, the innovation that first made Google famous — but this just shows how out of touch we are. PageRank is now just one of dozens of algorithms that Google uses to process search queries. Updates to the overall search system have been christened with a series of names, including Caffeine (2009), Hummingbird (2013), and Medic (2018). Since 2015, an AI learning system known as RankBrain has played a significant role. Its secrets are closely guarded intellectual property, so we don't know exactly how RankBrain works — but for all the reasons just discussed, it's quite likely that Google engineers don't know either. Furthermore, Google's near-monopoly on search makes Google rankings so important to the visibility and sales of other firms that a whole industry of "search engine optimization" (SEO) has arisen to analyze how the algorithm evaluates search queries; they then advise firms on how to tweak their websites to raise their pages' search rankings. Large parts of the web are thus constantly evolving in response to the recommendations of these SEO firms, creating a continuous feedback cycle with Google's search algorithms. Similar feedback loops mark YouTube, Twitter, Facebook, and all other major platforms.

The many scandals surrounding the 2016 US presidential election, including "fake news," Russian disinformation, resurgent white supremacism, and the Cambridge Analytica episode, have forced major platforms to police their content more closely. The nature of these responses shows the limits of algorithmic agency. In 2017, Google hired some 10,000 "quality raters" to identify "upsetting or offensive" content such as Holocaust-denial and white-supremacist websites. Engineers then adjust Google's search algorithms so as to lower the ranking of those results — or perhaps, as in the "gorilla" case, block them altogether [14]. Facebook and Twitter have also hired large numbers

of people to review and remove false and offensive content. Faced with a relentless onslaught of clever bots, as well as credulous and/or malevolent human users, the success of these strategies has been partial at best.

As I said earlier, it's hard to find a soundbite phrase to capture all this. Maybe it's a **principle of framing feedbacks:** that comprehending algorithmic systems requires backing out from a narrow focus on algorithms and data per se to a broader frame that encompasses some of the feedback loops I've described, or at the very least takes seriously the anthropology of human software developers [23]. Maybe it's a **principle of infrastructure** [19]: as my colleagues and I have articulated elsewhere, many modern infrastructures are not systems at all, but rather complex clusters of interacting elements best captured by organic or ecological metaphors [8]. Or it might be called the **principle of hybrid agency**, focusing on the constant interplay of individual choice, social norming, organizational change, and algorithmic action.

To capture all three of these notions at once, we could call it a **principle of infrastructuration**, my bad intellectual pun on Giddens' structuration theory [12; 13]. (Sorry about that.) To paraphrase Giddens, infrastructure shapes, limits, and enables agency. Meanwhile, agents (now including algorithmic agents) constantly *perform* infrastructure, constantly regenerating it but also transforming it over time. The most appropriate moniker of all might be **the principle of Borgian assimilation.** As the hivemind cyborg aliens from *Star Trek* put it, "Lower your shields and surrender your ships. We will add your biological and technological distinctiveness to our own. Your culture will adapt to service us. Resistance is futile."[6]

We still lack adequate intellectual tools to engage this fast-changing, globally active Borgian world. Notions such as sociomateriality, hybrid agency, monsters, and cyborgs take us in the right direction, but seem too simple and impoverished to capture the rich complexity of these interactions. Methods such as algorithm audits can help us understand how algorithmic systems may create or reinforce undesirable biases, but draw the frame too narrowly. Statisticians and practitioners can help us understand how biased and undesirable outcomes can occur even when designers do everything possible to eliminate them, and even in the presence of much more and much better data about human populations than we have ever had before [1; 16]. Behavioral economics and cognitive psychology remind us how little we really know our own minds [17; 18], how much we overestimate our rationality and the quality of our perception, and how embarrassingly susceptible we all are to subtle influences that can now be amplified and disseminated widely by algorithmic systems.[7]

We have been assimilated, all too willingly, and there is probably no going back.

---

[6] As articulated by the Borg in *Star Trek: First Contact* (1996).

[7] One of the best discussions of these susceptibilities and how algorithmic systems can manipulate them is an astonishingly direct 2016 presentation by Alexander Nix of Cambridge Analytics, "The Power of Big Data and Psychographics," available at www.youtube.com/watch?v=n8Dd5aVXLCc.

References

1. Barocas S, Selbst AD (2016) Big data's disparate impact. California Law Review 104:671–732. http://www.californialawreview.org/wp-content/uploads/2016/06/2Barocas-Selbst.pdf
2. Bessi A, Ferrara E (2016) Social bots distort the 2016 US Presidential election online discussion. First Monday 21. http://uncommonculture.org/ojs/index.php/fm/article/view/7090/5653
3. Boyd D, Marwick AE, Levey K (2014) The Networked Nature of Algorithmic Discrimination. Open Technology Institute, https://www.danah.org/papers/2014/DataDiscrimination.pdf
4. Burrell J (2016) How the machine 'thinks': Understanding opacity in machine learning algorithms. Big Data & Society 3:1–12. doi 10.1177/2053951715622512.
5. Dietvorst BJ, Simmons JP, Massey C (2015) Algorithm aversion: people erroneously avoid algorithms after seeing them err. Journal of Experimental Psychology: General 144:114–126. doi 10.1037/xge0000033.
6. Domingos P (2012) A few useful things to know about machine learning. Communications of the ACM 55:78–87. doi 10.1145/2347736.
7. Edwards PN (2010) A Vast Machine: Computer Models, Climate Data, and the Politics of Global Warming. MIT Press, Cambridge.
8. Edwards PN, Jackson SJ, Bowker GC, Knobel CP (2007) Understanding Infrastructure: Dynamics, Tensions, and Design. Deep Blue, Ann Arbor. http://hdl.handle.net/2027.42/49353
9. Eslami M, Karahalios K, Sandvig C, Vaccaro K, Rickman A, Hamilton K, Kirlik A (2016) First I "like" it, then I hide it. the 2016 CHI Conference:2371–2382. doi 10.1145/2858036.
10. Galloway AR (2006) Gaming: Essays on algorithmic culture. University of Minnesota Press, Minneapolis.
11. Galloway AR (2014) The Cybernetic Hypothesis. differences 25:107–131. doi 10.1215/10407391-2420021.
12. Giddens A (1981) Agency, Institution, and Time-space Analysis. In: Knorr-Cetina K, Cicourel AV (eds) Advances in Social Theory and Methodology: Toward an Integration of Micro- and Macro-sociologies. Routledge & Kegan Paul, Boston.
13. Giddens A (1984) The Constitution of Society. Spectrum Educational Enterprises, Washington, D.C.
14. Guynn J (2017) Google starts flagging offensive content in search results. USA Today. https://www.usatoday.com/story/tech/news/2017/03/16/google-flags-offensive-content-search-results/99235548/
15. Hallinan B, Striphas T (2015) Recommended for you: The Netflix Prize and the production of algorithmic culture. New Media & Society 18:117–137. doi 10.1177/1461444814538646.

16. Hardt M (2014) How big data is unfair: understanding unintended sources of unfairness in data driven decision making. Medium. https://medium.com/@mrtz/how-big-data-is-unfair-9aa544d739de
17. Kahneman D (2003) Maps of bounded rationality: Psychology for behavioral economics. American Economic Review 93:1449–1475. http://www.jstor.org/stable/3132137
18. Kahneman D (2011) Thinking, Fast and Slow. Macmillan, New York.
19. Plantin J-C, Lagoze C, Edwards PN, Sandvig C (2016) Infrastructure studies meet platform studies in the age of Google and Facebook. New Media & Society 10:1–18. doi 10.1177/1461444816661553.
20. Rader E, Gray R (2015) Understanding User Beliefs About Algorithmic Curation in the Facebook News Feed. the 33rd Annual ACM Conference:173–182. doi 10.1145/2702123.
21. Royal Society (2017) Machine learning: the power and promise of computers that learn by example. Royal Society, London.
22. Schüll ND (2012) Addiction by design: Machine gambling in Las Vegas. Princeton University Press, Princeton.
23. Seaver N (2018) What should an anthropology of algorithms do? Cultural Anthropology 33:375–385. https://culanth.org/articles/966-what-should-an-anthropology-of-algorithms-do
24. Striphas T (2015) Algorithmic culture. European Journal of Cultural Studies 18:395–412. doi 10.1177/1367549415577392.
25. Vosoughi S, Roy D, Aral S (2018) The spread of true and false news online. Science 359:1146–1151. doi 10.1126/science.aap9559.
26. Wallach H (2014) Big Data, Machine Learning, and the Social Sciences. Medium. http://medium.com/@hannawallach/big-data-machine-learning-and-the-social-sciences-927a8e20460d