CHAPTER 2

Getting Past Logic

In law, as we saw in Chapter 1, the contrast between formalism and inductivism was evident practically from the moment jurists began to consider that logic might not explain everything about the law. The contrast continues to define lines that run through legal studies and judicial politics, especially in the United States and also to an extent in other common law jurisdictions. The lines are readily discernible. Volumes of literature and on-going disputes are gathered on one side or the other. People who think about and practice law have identified themselves in adversarial terms by reference to which side of those lines they stand on. In computing, as we also noted in Chapter 1, the lines are not drawn in quite such clear relief. They are certainly not the reference point for the intellectual identity of opposing camps of computer scientists. The conceptual shift that underpins the emergence of machine learning has had enormous impact, but it has not been an object of sustained discourse, much less of pitched ideological battle. Computer scientists thus, perhaps, enjoy a certain felicity in their professional relations, but they also are probably less alert to the distinction that machine learning introduces between their present endeavors and what they were doing before.

We will examine in the three chapters immediately after this the ingredients that go into making machine learning so different from traditional algorithm-based programming. The ingredients are data (Chapter 3), pattern finding (Chapter 4), and prediction (Chapter 5). Before examining these, we wish to consider further the contrast between machine learning and what came before. The rise of legal realism was explicitly a challenge

T. D. Grant and D. J. Wischik, *On the path to AI*,
https://doi.org/10.1007/978-3-030-43582-0_2

to what came before in law, and, so, the contrast was patent. With the emergence of induction-driven machine learning, the contrast ought to be no less clear, but people continue to miss it. Getting past logic is necessary, if one is to get at what's new about machine learning—and at why this kind of computing presents special challenges when it comes to values in society at large.

2.1 Formalism in Law and Algorithms in Computing

Legal formalists start with the observation, to which few would object, that law involves rules. They identify the task in law, whether performed by a lawyer or a judge, to be that of applying the rules to facts, again in itself an unobjectionable, or at least unremarkable, proposition. Where formalism is distinctive is in its claim that these considerations supply a complete understanding of the law. "Legal reasoning," said the late twentieth century critical legal scholar Roberto Unger, "is formalistic when the mere invocation of rules and deduction of conclusions from them is believed sufficient for every authoritative legal choice."[1] An important correlate follows from such a conception of the law.[2] The formalists say that, if the task of legal reasoning is performed correctly, meaning in accordance with the logic of the applicable rules, the lawyer or judge reaches the correct result. The result might consist in a judgment (adopted by a judge and binding on parties in a dispute) or in a briefing (to her client by a lawyer), but whatever the forum or purpose, the result comes from a logical operation, not differing too much from the application of a mathematical axiom. In the formalists' understanding, it thus follows that the answer given to a legal question, whether by a lawyer or by a judge, is susceptible of a logical process of review. An erroneous result can be identified by tracing back the steps that the lawyer or judge was to have followed and finding a step in the operation where that technician made a mistake. Why a correct judgment is correct thus can be explained by reference to the rules and reasoning on which it is based; and an incorrect one can be diagnosed much the same way.

In Oliver Wendell Holmes, Jr.'s day, though legal formalism already had a long line of distinguished antecedents such as Blackstone (whom we quoted in our Prologue), one contemporary of Holmes, C. C. Langdell, Dean and Librarian of the Harvard Law School, had come to be specially associated with it. Holmes himself identified the Dean as arch-exponent of this mode of legal reasoning. In a book review in 1880, he referred to Langdell, who was a friend and colleague, as "the greatest living legal theologian."[3] The compliment was a back-handed one when spoken among self-respecting rationalists in the late nineteenth century. In private correspondence around the same time, Holmes called Langdell a jurist who "is all for logic and hates any reference to anything outside of it."[4] A later scholar, from the vantage of the twenty-first century, has suggested that Langdell was less a formalist than Holmes and others made him out to be but nevertheless acknowledges the widespread association and the received understanding: "[l]egal formalism [as associated with Langdell] consisted in the view that deductive inference from objective, immutable legal principles determines correct decisions in legal disputes."[5]

Whether or not Langdell saw that to be the only way the law functions, Holmes certainly did not, and the asserted contrast between the two defined lines which remain familiar in jurisprudence to this day. In his own words, Holmes rejected "the notion that the only force at work in the development of the law is logic."[6] By this, he did *not* mean that "the principles governing other phenomena [do not] also govern the law."[7] Holmes accepted that logic plays a role in law: "[t]he processes of analogy, discrimination, and deduction are those in which [lawyers] are most at home. The language of judicial decision is mainly the language of logic."[8] That deductive logic and inductive reasoning co-exist in law may already have been accepted, at least to a degree, in Holmes's time.[9] What Holmes rejected, instead, was "the notion that a [legal system]… can be worked out like mathematics from some general axioms of conduct."[10] It was thus that Holmes made sport of a "very eminent judge" who said "he never let a decision go until he was absolutely sure that it was right" and of those who treat a dissenting judgment "as if it meant simply that one side or the other were not doing their sums right, and if they would take more trouble, agreement inevitably would come."[11] If the strict formalist thought that all correctness and error in law are readily distinguished and their points of origin readily identified, then Holmes thought that formalism as legal theory was lacking.

The common conception of computer science is analogous to the formalist theory of law. Important features of that conception are writing a problem description as a formal specification; devising an algorithm, i.e. a step-by-step sequence of instructions that can be programmed on a computer; and analyzing the algorithm, for example to establish that it correctly solves the specified problem. "The term *algorithm* is used in computer science to describe a … problem-solving method suitable for implementation as a computer program. Algorithms are the stuff of computer science: they are central objects of study in the field."[12] In some areas the interest is in devising an algorithm to meet the specification. For example, given the problem statement *Take a list of names and sort them alphabetically*, the computer scientist might decompose it recursively into *to sort a list, first sort the first half, then sort the second half, then merge the two halves*, and then break these instructions down further into elementary operations such as *swap two particular items in the list*. In other areas the interest is in the output of the algorithm. For example, given the problem statement *Forecast the likely path of the hurricane*, the computer scientist might split a map into cells and within each cell solve simple equations from atmospheric science to predict how wind speed changes from minute to minute. In either situation, the job of the computer scientist is to codify a task into simple steps, each step able to be (i) executed on a computer, and (ii) reasoned about, for example to debug why a computer program has generated an incorrect output (i.e. an incorrect result). The steps are composed in source code, and scrutinizing the source code can disclose how the program worked or failed. Success and failure are ready to see. The mistakes that cause failure, though sometimes frustratingly tangled in the code, are eventually findable by a programmer keen enough to find them.

2.2 Getting Past Algorithms

Machine learning however neither works like algorithmic code nor is to be understood as if it were algorithmic code. Outputs from machine learning are not effectively explained by considering only the source code involved. Kroll et al., whom we will consider more closely in a moment, in a discussion of how to make algorithms more accountable explain:

> Machine learning… is particularly ill-suited to source code analysis because it involves situations where the decisional rule itself emerges automatically from the specific data under analysis, sometimes in ways that no human can explain. In this case, source code alone teaches a reviewer very little, since the code only exposes the machine learning method used and not the data-driven decision rule.[13]

In machine learning, the job of the computer scientist is to assemble a training dataset and to program a system that is capable of learning from that data. The outcome of training is a collection of millions of fine-tuned parameter values that configure an algorithm. The algorithms that computer scientists program in modern machine learning are embarrassingly simple by the standards of classic computer science, but they are enormously rich and expressive by virtue of their having millions of parameters.

The backbone of machine learning is a simple method, called *gradient descent*.[14] It is through gradient descent that the system arrives at the optimum settings for these millions of parameters. It is how the system achieves its fine-tuning. To be clear, it is not the human programmer who fine tunes the system; it is a mathematical process that the human programmer sets in motion that does the fine-tuning. Thus built on its backbone of gradient descent, machine learning has excelled at tasks such as image classification and translation, tasks where formal specification and mathematical logic have not worked. These achievements justify the encomia that this simple method has received. "Gradient descent can write code better than you."[15] After training, i.e. after configuring the algorithm by setting its parameter values, the final stage is to invoke the algorithm to make decisions on instances of new data. It is an algorithm that is being invoked, in the trivial sense that it consists of simple steps which can be executed on a computer; but its behavior cannot be understood by reasoning logically about its source code, since its source code does not include the learnt parameter values.

Moreover, it is futile to try to reason logically about the algorithm even given all the parameter values. Such an analysis would be as futile as analyzing a judge's decision from electroencephalogram readings of her brain. There are just too many values for an analyst to make sense of. Instead, machine-learnt algorithms are evaluated empirically, by measuring how they perform on test data. Computer scientists speak of "black-box" and "white-box" analysis of an algorithm. In white-box analysis we consider the internal structure of an algorithm, whereas in black-box analysis we consider only its inputs and outputs. Machine-learnt algorithms are evaluated purely on the basis of which outputs they generate for which inputs, i.e. by black-box analysis. Where computer scientists have sought to address concerns about discrimination and fairness, they have done so with black-box analysis as their basis.[16] In summary, a machine learning "algorithm" is better thought of as an opaque embodiment of its training dataset and evaluation criterion, not as a logical rules-based procedure. Problems with which machine learning might be involved (such as unfair discrimination) thus are not to be addressed as if it were a logical rules-based procedure.

2.3 The Persistence of Algorithmic Logic

Yet people continue to address machine learning as if it were just that—a logical rules-based procedure not different in kind from traditional computer programming based on algorithmic logic. This inadequate way of addressing machine learning—addressing it as though the source code of an algorithm is responsible for producing the outputs—is not limited to legal discourse. It is however very much visible there. Formal, algorithmic descriptions of machine learning are ubiquitous in legal literature.[17] The persistence of algorithmic logic in descriptions of how computers work is visible even among legal writers who otherwise acknowledge that machine learning is different.[18]

Even Kroll et al., who recognize that machine learning "is particularly ill-suited to source code analysis," still refer to "a machine [that] has been 'trained' through exposure to a large quantity of data and *infers a rule from the patterns it observes.*"[19] To associate machine learning with "a rule from the patterns it observes" will lead an unwary reader to conclude that the machine has learnt a cleanly stated rule in the sense of law or of decision trees. In fact, the machine has done no such thing. What it has done is find a pattern which is "well beyond traditional interpretation," these being the much more apt words that Kroll et al. themselves use to acknowledge the opacity of a machine learning mechanism.[20]

Kroll and his collaborators have addressed at length the challenges in analyzing the computer systems on which society increasingly relies. We will turn in Chapters 6–8 to extend our analogy between two revolutions to some of the challenges. A word is in order here about the work of Kroll et al., because that work highlights both the urgency of the challenges and the traps that the persistence of algorithmic logic presents.

There are many white-box tools for analyzing algorithms, for example based on mathematical analysis of the source code. Kroll et al. devote the bulk of their paper on *Accountable Algorithms* (2017) to white-box software engineering tools and to the related regulatory tools that can be used to ensure accountability. The persistence of algorithmic logic here, again, may lead to a trap: the unwary reader thinks machine learning algorithms are per se algorithms; here are tools for making algorithms accountable; *therefore we can make machine learning accountable*. Kroll et al. mark the trap with a warning flag, but it is a rather small one. They concede in a footnote that all the white-box techniques they discuss simply do not apply to machine learning mechanisms and that the best that can be done is to regulate the *decision* to use machine learning: “Although some machine learning systems produce results that are difficult to predict in advance and well beyond traditional interpretation, the choice to field such a system instead of one which can be interpreted and governed is itself a decision about the system’s design.”[21] This isn’t saying how to understand machine learning better. It’s saying not to use machine learning. The result, if one were to follow Kroll et al., would be to narrow the problem set to a much easier question—what to do about systems that use only traditional algorithmic logic.

Indeed, it is the easier question that occupies most of Kroll et al.’s *Accountable Algorithms*. Forty-five pages of it discuss white-box analysis that doesn’t apply to machine learning systems. Eighteen pages then consider black-box analysis. An exploration of black-box analysis is more to the point—the point being to analyze machine learning.

But a trap is presented there as well. The pages on black-box analysis are titled “Designing algorithms to ensure fidelity to substantive policy choices.” Black-box analysis by definition is agnostic as to the design of the “algorithms” that are producing the results it analyzes. Black-box analysis is concerned with the output of the system, not with the inner workings that generate the output. To suggest that “[d]esigning algorithms” the right way will “ensure fidelity” to some external value is to fall into the algorithmic trap. It is to assume that algorithmic logic is at work, when the real challenge involves machine learning systems, the distinctive feature of which is that they operate outside that logic. Black-box analysis, which Kroll et al. suggest relies upon the design of an algorithm, in fact works just as well in analyzing decisions made by a flock of dyspeptic parrots.

Kroll in a later paper expands the small flag footnote and draws a distinction between systems and algorithms.[22] As Kroll uses the words, the *system* includes the human sociotechnical context—the power dynamics and the human values behind the design goals, and so on—whereas the *algorithm* is the technical decision-making mechanism embedded in the system. It is the “system,” in the sense that he stipulates, that mainly concerns him.[23] Kroll argues that a machine learning “system” is necessarily scrutable, since it is a system built by human engineers, and human choices can always be scrutinized. But, once more, this is an observation that would apply just as much if the engineers' choice was to use the parrot flock. It is the essence of the black-box that we know only what output it gives, whether a name, a color, a spatial coordinate, or a squawk. We don't know how it arrived at the output. In so far as Kroll addresses machine learning itself, he does not offer tools to impart scrutability to it but, instead, only this: "the question of how to build effective white-box testing regimes for machine learning systems is far from settled."[24] To say that white-box testing doesn't produce "settled" answers to black-box questions is an understatement. And the problem it understates is the very problem that activists and political institutions are calling on computer scientists to address: how to test a machine learning system to assure that it does not have undesirable effects. What one is left with, in talking about machine learning this way, is a hope: namely, a hope that machine learning, even though it may be the most potent mechanism yet devised for computational operations, will not be built into "systems" by the many individuals and institutions who stand to profit from its use. Exploration of white-box, logical models of scrutability reveals little or nothing about machine learning. Insistence on such exploration only highlights the persistence of algorithmic logic notwithstanding the revolution that this technology represents.

Many accounts of machine learning aimed at non-specialists display these shortcomings. Lawyers, as a particular group of non-specialist, are perhaps particularly susceptible to misguided appeals to logical models of computing. It is true that statutory law has been compared to computer source code[25]; lawyers who are at heart formalists may find the comparison comforting.[26] It is also true that an earlier generation of software could solve certain fairly hard legal problems, especially where a statutory and regulatory regime, like the tax code, is concerned. Machine learning, however, is not limited in its applications to tasks that, like calculating a tax liability, are themselves algorithmic (in the sense that a human operator can readily describe the tasks as a logical progression applying fixed rules to given facts). Computer source code is not the characteristic of machine learning that sets it apart from the kind of computing that came before. Lawyers must let go the idea that logic—the stepwise deduction of solutions by applying a rule—is what's at work in the machine learning age. Herein, we posit, reading Holmes has a salutary effect.

Holmes made clear his position by contrasting it against that taken by jurists of what we might, if anachronistically, call the algorithmic school, that is to say the formalists. In *Lochner v. New York*, perhaps his most famous dissent, Holmes stated, "General propositions do not decide concrete cases."[27] This was to reject deductive reasoning in plain terms and in high profile. Whether or not we think that is a good way to think about law,[28] it is precisely how we must think if we are to understand machine learning; machine learning demands that we think beyond logic. Computer scientists themselves, as much as lawyers and other laypersons, ought to recognize this conceptual transition, for it is indispensable to the emergence of machine learning which is now transforming their field and so much beyond.

With the contrast in mind between formal ways of thinking about law and about computing, we now will elaborate on the elements of post-logic thinking that law and computing share: data, pattern finding, and prediction.

Notes

1. Unger (1976) 194.
2. Unger's description is not the only or necessarily the definitive description of legal formalism. A range of nuances exists in defining legal formalism. Unger's pejorative use of the word "mere," in particular, may well be jettisoned: to say that logical application of rules to facts is the path to legal results by no means entails that the cognitive operations involved—and opportunities for disagreement—are trivial. Dijkstra's caution that only very good mathematicians are ready for the logical rigors of (traditional) computing merits an analogue for lawyers.
3. Holmes, *Book Notice Reviewing a Selection of Cases on the Law of Contracts, with a Summary of the Topics Covered by the Cases, By C.C. Langdell*, 14 Am. L. Rev. 233, 234 (1880).
4. Letter from Holmes to Pollock (Apr. 10, 1881), reprinted De Wolfe Howe (ed.) (1942) 17. Though Holmes's skepticism over the formalist approach most famously targeted Langdell, Walbridge Abner Field, Chief Justice of the Massachusetts Supreme Judicial Court, also fell into the sights. See Budiansky (2019) 199–200.
5. Kimball (2009) 108 and references to writers id. at 108 n. 134. See also Cook 88 N.D. L. REV. 21 (2012); Wendel, 96 Corn. L. Rev. 1035, 1060–65, 1073 (2011). Langdell's undoubted contribution—the case method of teaching law—provoked rebellion by his students for the very reason that it did *not* start with clear rules: see Gersen, 130 Harv. L. Rev. 2320, 2323 (2017). Cf. Grey, *Langdell's Orthodoxy*, 45 U. Pitt. L.

REV. 1 (1983), who drew attention to the rigor and advantages of classic formalism.

Richard Posner, to whose understanding of Holmes's legal method we will return later below (Chapter 10, p. 119), describes logic as central to legal reasoning of the time, though "common sense" was also involved:

> The task of the legal scholar was seen as being to extract a doctrine from a line of cases or from statutory text and history, restate it, perhaps criticize it or seek to extend it, all the while striving for 'sensible' results in light of legal principles and common sense. Logic, analogy, judicial decisions, a handful of principles such as *stare decisis*, and common sense were the tools of analysis.

Posner, 115(5) HARV. L. REV. 1314, 1316 (2002).

6. 10 HARV. L. REV. at 465.
7. Id. For a consideration of the role of Langdell's logic in Holmes's conception of law, see Brown & Kimball, 45 AM. J. LEG. HIST. 278–321 (2001).
8. 10 HARV. L. REV. at 465–66.
9. Which is not to say that Holmes necessarily would have agreed with other jurists' understanding of *how* they co-exist. Edward Levi, in his text on legal reasoning, said this: "It is customary to think of case-law reasoning as inductive and the application of statutes as deductive," Levi, AN INTRODUCTION TO LEGAL REASONING (1949) 27. Common law courts have described their function in applying rules derived from past cases as involving both kinds of reasoning—induction to derive the rules; deduction from a rule thus derived. See, e.g., Skelton v. Collins (High Court of Australia, Windeyer J., 1966) 115 CLR 94, 134; Home Office v. Dorset Yacht Co Ltd. [1970] A.C. 1004, 1058–59 (House of Lords, Diplock LJ). Cf. Norsk Pacific Steamship et al. v. Canadian National Railway Co., 1992 A.M.C. 1910, 1923 (Supreme Court of Canada, 1992, McLachlin J.); In the Matter of Hearst Corporation et al. v. Clyne, 50 N.Y.2d 707, 717, 409 N.E.2d 876, 880 (Court of Appeals of New York, 1980, Wachtler J.). Cf. Benjamin Cardozo, THE NATURE OF THE JUDICIAL PROCESS (1921) 22–23. Levi's contrast—between induction for the common law and deduction for rules such as statutes contain—has been noted in connection with computer programming: see, e.g., Tyree (1989) 131.
10. 10 HARV. L. REV. at 465.
11. Id.
12. Robert Sedgewick, ALGORITHMS (4th ed.) (2011) Section 1.
13. Kroll et al., *Accountable Algorithms*, U. PA. L. REV. at 638 (2017).
14. For a formal description of gradient descent, and some of its variants, see e.g. Bishop, PATTERN RECOGNITION AND MACHINE LEARNING

(2007) Section 5.2.4. Gradient descent is attributed to the French mathematician Augustin-Louis Cauchy (1789–1857) in a work published in 1847 (see Claude Lemaréchal, *Cauchy and the Gradient Method* (2012), Documental Math. p. 251). Gradient descent when applied to finding parameter values for a neural network is known as *back propagation*.

15. Tweet by @karpathy, 1.56 p.m., Aug. 4, 2017. https://twitter.com/karpathy/status/893576281375219712? See further, e.g., Murphy (2012) 247, 445.
16. Kroll et al., *supra* n. 76, at 650–51, 685–87. The tests for discrimination in machine learning algorithms described there, which come from Cynthia Dwork et al., *Fairness Through Awareness*, ITCS Conf. Proc. (3rd) (2012), derive from a black-box definition.
17. See for example works cited by Barocas & Selbst, *Big Data's Disparate Impact*, 104 Cal. L. Rev. 671, 674 n. 11. See also Alan S. Gutterman, *Glossary of Computer Terms*, in Business Transactions Solutions § 217:146 (Jan. 2019 update) ("the machine merely follows a carefully constructed program"); Chessman, 105 Cal. L. Rev. 179, 184 (2017) ("Evidence produced by computer programs arguably merits additional scrutiny… because the complexity of computer programs makes it difficult… to detect errors"); Gillespie in Gillespie et al. (eds.) (2014) 167, 192 ("algorithmic logic… depends on the proceduralized choices of a machine designed by human operators to automate some proxy of human judgment"), and the same as quoted by Carroll, *Making News: Balancing Newsworthiness and Privacy in the Age of Algorithms*, 106 Geo. L.J. 69, 95 (2017). As we will observe below, algorithmic logic persists in legislation and regulation as well: see Chapter 6, p. 70.
18. See for example Roth, 126 Yale L. J. 1972, 2016–2017 (2017) who says, "[A] machine's *programming*… could cause it to utter a falsehood by design" (emphasis added). Another example is found in Barocas & Selbst, 104 Cal. L. Rev. at 674 (2016), who, like Roth, understand that "the data mining process itself" is at the heart of machine learning, but still say that "[a]lgorithms could exhibit these [invidious] tendencies even if they have not been manually programmed to do so…", a formulation which accounts for programs that are not written in the conventional way, but which still does not escape the gravity of the idea of source code (i.e., that which is "programmed"). Benvenisti has it on the mark, when he cites Council of Europe Rapporteur Wagner's observation that "provision of all of the source codes… may not even be sufficient." Benvenisti, *Upholding Democracy and the Challenges of New Technology: What Role for the Law of Global Governance?* 29 EJIL 9, 60 n. 287 (2018) quoting, inter alia, Wagner, Rapporteur, Committee of Experts on Internet Intermediaries (MSI-NET, Council of Europe), *Study on the Human Rights Dimensions of Algorithms* (2017) 22.

19. Kroll et al., *supra* n. 76, at 679 (emphasis added).
20. Id., n. 9. Others have used the same terms, similarly running the risk of conflating the output of machine learning systems with "rules." For example distinguished medical researchers Ziad Obermeyer & Ezekiel J. Emanuel, who have written cogent descriptions of how machine learning works, refer to machine learning as "approach[ing] problems as a doctor progressing through residency might: *by learning rules from data*" (emphasis added). Obermeyer & Ezekiel, *Predicting the Future—Big Data, Machine Learning, and Clinical Medicine*, 375(13) NEJM 1216 (Sept. 29, 2016).
21. Kroll et al. at n. 9. The proviso "*some* machine learning systems" acknowledges that at least some machine learning systems such as neural networks are well beyond traditional interpretation. True, some simple systems labeled "machine learning" produce easily understood rules and are therefore amenable to white-box analysis. One hears sales pitches that use the term "machine learning" to refer to basic calculations that could be performed in a simple Excel spreadsheet. These are not the systems making the emerging machine learning age, or its challenges, distinctive.
22. Kroll (2018) PHIL. TRANS. R. SOC. A 376.
23. Kroll (2018) indeed is about systems not algorithms—e.g., it uses "algorithm" and related words 16 times, and "system" and related words 235 times.
24. Id., p. 11. There have been attempts to find white-box approaches for understanding neural networks, notably Shwartz-Ziv & Tishby (2017), but such theory is not widely accepted: Saxe et al. (2018).
25. See esp. Lessig's extended analogy: Lessig, CODE AND OTHER LAWS OF CYBERSPACE (1999) 3–8, 53. See also Tyree (1989) 131.
26. For an analogy from psychology positing that legal formalism comes before supposedly "later stages of cognitive and moral development," see Huhn, 48 VILL. L. REV. 305, 318–39 (2003). We venture no opinion as to the whether Huhn's analogy is convincing, either for law or for computer science.
27. *Lochner v. New York* , 198 U.S. 45, 76 (1905) (Holmes, J., dissenting).
28. Posner observes jurists have disputed formalism for two thousand years: Posner (1990) 24–25. The contrast between formalism and other types of legal analysis, whenever the latter arose and whatever name they go by ("positivism," "realism," etc.), is at the heart of much of the intellectual—and ideological—tension in American lawyering, judging, and law scholarship. It is visible in international law as well. See Prologue, pp. xii–xiii, n. 17. As we stated in Chapter 1, we address the contrast for purposes of analogy to machine learning, not to join in a debate over the merits of formalism or its alternatives in law.