Available online at www.sciencedirect.com
ScienceDirect
Artificial Intelligence 170 (2006) 1222–1226
Artificial Intelligence
www.elsevier.com/locate/artint

Book review

Marcus Hutter, Universal Artificial Intelligence, Springer, 2004.

Tim Oates [a,*], Waiyian Chong [b]

[a] *Department of Computer Science and Electrical Engineering, University of Maryland, Baltimore County, 1000 Hilltop Circle, MD 21250, USA*
[b] *Department of Computer Science, University of Maryland, College Park, MD 20742, USA*

Available online 7 November 2006

In the early days of the field there was considerable optimism about the near-term prospects for constructing artificially intelligent systems. In 1957, Herbert Simon said "there are now in the world machines that think, that learn and that create", and "in a visible future ... the range of problems they can handle will be coextensive with the range to which the human mind has been applied" [6]. There is also the rumor that sometime in the mid-sixties an MIT undergrad was tasked with solving the object recognition problem as a summer project [1]. What we've learned since then is that the AI problem in its entirety and many of its subproblems are indescribably hard.

Nevertheless, there have been some stunning successes, such as IBM's Deep Blue that plays chess as well as the best human players. However, these successes are always obtained in restricted domains that are by no means "coextensive with the range to which the human mind has been applied". As Drew McDermott said, "Deep Blue is unintelligent because it is so narrow. It can win a chess game, but it can't recognize, much less pick up, a chess piece. It can't even carry on a conversation about the game it just won. Since the essence of intelligence would seem to be breadth, or the ability to react creatively to a wide variety of situations, it's hard to credit Deep Blue with much intelligence" [4].

The contribution of Marcus Hutter's book "Universal Artificial Intelligence" is a model, called AIXI, that is immune to criticisms such as McDermott's. The primary claim of the book is that AIXI is *universally optimal*. AIXI is *universal* because, regardless of the environment in which AIXI operates, it can "solve any solvable problem and learn any learnable task". In machine learning terms, AIXI has no parameters and is unbiased. It is *optimal* because there is no system that solves problems or learns tasks faster, i.e., with fewer interactions with the environment. The status of the claims to universality and to optimality in the book are different. Proofs of universality are offered, but support for the claim of optimality is less formally conclusive. In Hutter's own words, "we *expect* AIXI to be universally optimal" (italics not in the original). The book contains many results that are of interest independent of AIXI, particularly in Chapter 3 and the second half of Chapter 5, but reviewing these would take us too far afield.

Regardless of whether you are a card-carrying member of the Church of Strong AI, or you adhere to some other religion that argues for the impossibility of the existence of genuinely intelligent artificial systems, most people would treat claims such as these with a healthy dose of skepticism. Indeed, most people who make such claims are crackpots. But Marcus Hutter is by no means a crackpot. He has Ph.D. degrees in both Physics and Computer Science, a long list of publications, and a Kolmogorov medal. He is clearly passionate about the field of Artificial Intelligence, and sanguine about the prospects for solving The AI Problem soon. His web page says "I'm interested in problems on the boundary between science and philosophy which have a chance of being solved in my expected lifetime, especially Artificial Intelligence".

* Corresponding author.
E-mail address: oates@cs.umbc.edu (T. Oates).

doi:10.1016/j.artint.2006.10.005

Hutter is also taking unusual steps to advance the state of knowledge in the areas of AI that he deems most important. Proceeds from the sale of his book are being used to fund cash prizes for solving some of the problems it leaves open. For example, you can earn 500 Euros for constructing a universal semimeasure with posterior convergence for all Martin–Löf random sequences. He is also offering up to 50,000 Euros for anyone who can compress a 100 MB subset of Wikipedia to less than 18 MB, which is the current record. Hutter claims that the abilities to compress well and to act intelligently are closely related, and the prize is meant to spur work that will shed light on the validity of that claim.

Before discussing the provocative ideas contained in the book, a few words on the text of the book are warranted. First, Hutter is very careful to both state his assumptions and point out the limitations of AIXI. Regardless of how outlandish the main claim of the book may sound (i.e., that AIXI is universally optimal), we believe that within the context of his assumptions the claim is justified, with the caveat that one of AIXI's limitations will be unacceptable to many. Rather than reveal that limitation now and poison the well, we'll allow the reader to make up their own mind once they understand what Hutter has done.

The second thing potential readers of the book should know is that it is exceptionally dense. On many of the book's pages there are fewer words than mathematical formulas. Such rigor is necessary to back up such strong claims, but anyone who is not an expert in both algorithmic information theory and sequential decision theory will find the text to be very rough going. This is especially true of the introduction, which is typically a place where readers can get the main ideas without diving into the details. Hutter says the following of his introduction. "This Chapter represents a short tour through the book. It is not meant as a gentle introduction for novices, but as a condensed presentation of the most important concepts and results of the book". We fear that Hutter's exciting ideas will not reach as broad an audience as they deserve simply because their presentation asks too much of the reader. That said, we now turn to what we hope is an intuitive explanation of the core content of the book.

1. Universal sequence prediction

For those familiar with the work of Ray Solomonoff, one way to characterize what Hutter has done is to say that what Solomonoff did for the problem of induction, Hutter has done for the problem of sequential decision making. For those unfamiliar with that work, read on.

Suppose you are trying to predict something, like the weather or the stock market. You have access to historical data and want to predict as accurately as possible whether it will be sunny tomorrow, or whether the market will close up or down tomorrow. Now imagine encoding the historical data as a bit string, and that the system producing the data is doing so according to probability distribution $\mu(x)$, which is the probability that the observed bit string starts with string x. Given that you've observed x, what is the best prediction for the next bit that will appear? If μ is known, you can compute $p(0|x)$ as $\mu(0x)/\mu(x)$, and likewise for $p(1|x)$.

But what if μ is unknown? William of Ockham said you should use the simplest model that is consistent with the data. Epicurus said you should use all of the models that are consistent with the data. And David Hume said you should find something else to do with your time, because there is no rational basis for induction [2]. Fortunately, Solomonoff didn't listen to Hume, and he invented the *universal prior* [7,8]. Let p be a program of length $l(p)$ and U be a universal Turing machine.[1] Let $U(p) = x*$ denote the fact that when U runs program p the output produced by U starts with x. The universal prior is defined as follows:

$$M(x) = \sum_{p:\, U(p)=x*} 2^{-l(p)}$$

That is, to compute $M(x)$ one must enumerate all programs p that, when run on a universal Turing machine, produce output that starts with x.

Note that $M(x)$ is similar to $\mu(x)$. Both assign probabilities to string prefixes. Whereas μ in some sense defines the environment producing the strings, M assigns probabilities based on the output of literally random programs. If a program p produces output that starts with x, that program contributes to $M(x)$. The amount that it contributes is

[1] To be precise, we should say that U is a universal monotone Turing machine. However, we will sacrifice some precision in the remainder of this review in an effort to convey intuitions more effectively.

an exponentially decaying function of its length. That is, if there are very simple (i.e., short) programs that produce output starting with x, then $M(x)$ will be large, making William of Ockham happy. Even if there are very complex (i.e., long) programs that produce output starting with x, they get to contribute to $M(x)$, albeit to a much smaller degree, making Epicurus happy.

Why should anyone other than William of Ockham and Epicurus care about $M(x)$? Because, regardless of what distribution, μ, governs the environment, as the length of x grows, $M(x)$ converges to $\mu(x)$. That is, with absolutely no prior knowledge (i.e., no bias) about μ, using M is eventually as good as using μ to make predictions. It is in this sense that Solomonoff's prior is *universal*. In fact, it can be shown that convergence is rapid and that loss bounds are tight.

Those familiar with the various "no free lunch" (NFL) theorems in machine learning [11] and optimization [12] may be skeptical of any claims to universality, including those of Solomonoff and Hutter. NFL theorems are invariably based on the assumption that the domain in which one is trying to learn can be computing *any* function. Algorithmic complexity suggests that domains characterized by long programs are uninteresting because they contain little structure about which one can learn. For example, consider two programs, one that produces a string of one million zeroes and another that produces a string of one million truly random bits. The former program is clearly much smaller than the latter, and thus the output of the former is easier to predict than the output of the latter. By analogy, those in the camp of William of Ockham, and thus the camps of Solomonoff and Hutter, would argue that any domain for which $l(p)$, the length of the program the domain is "running", is large is uninteresting from a learning standpoint precisely because it contains little structure, and the universal prior does the right thing by assigning it little weight.

Given the above results, why is anyone still working on induction? For the simple reason that $M(x)$ is not computable because it involves determining which programs, out of all possible programs, cause a universal Turing machine to produce output that starts with a given string. This will be the fly in Hutter's ointment as well.

2. Universal sequential decisions

Rather than being interested in sequential prediction problems, Hutter is interested in sequential decision problems. Given an agent that at time t can take action $y_t \in Y$ which has some effect on the environment and results in observation $x_t \in X$ and scalar reward $r_t \in [0, r_{\max}]$, the goal of the agent is to learn to choose actions that maximize future rewards. A typical assumption in such cases is that the environment is Markovian, that the effects of the agent's actions depend only on the state of the environment at time t, not on the state at any earlier times. In contrast, Hutter is interested in real-world, partially observable, non-Markovian environments, where the effects of the agent's actions can depend on past states of the environment. This means that the agent's policy, which is used to choose actions based on observations, must also depend on arbitrarily old observations which convey some information about arbitrarily old states of the environment.

Furthermore, Hutter does not need to assume that the probability distributions governing observations and rewards are known. He deals with this lack of knowledge, which is typical of real-world domains, in precisely the same way that Solomonoff dealt with the problem of unknown μ for sequence prediction. Assume each observation, x_t, includes both the standard things one might observe, such as values produced by sensors at time t, and the reward received at time t. Let q be an environment—a *program*—that takes as input a string of actions, y, and produces as output a string of observations, x, that includes rewards. When this holds for a particular q, y, and x we write $q(y) = x$. The analog of Solomonoff's universal prior is then given by the following:

$$\xi(y, x) = \sum_{q:\, q(y)=x} 2^{-l(q)}$$

The symbol ξ is the Greek character xi, thus, apparently, the name AIXI.

Note the parallels between ξ and M. Observation sequences that can be explained by simple environments (i.e., can be produced by short programs given the action sequence) have high probability. More importantly, Hutter proves that as the length of the action/observation sequence tends to infinity, ξ converges to μ, the true but unknown distribution governing the behavior of the environment. Unfortunately, the parallels also include the fact that ξ is not computable, though we will return to the issue of computability at the end of this review.

Given increasingly accurate estimates of μ, via ξ, as the agent takes actions in the environment, the remaining task is to develop an algorithm for finding the optimal policy, the one that maximizes future rewards. In cases where this maximization is restricted to a finite horizon into the future, e.g., the next k actions, there is a simple algorithm for finding the optimal policy—enumerate all action sequences of length k and use ξ to compute their expected value, i.e., the one with the maximum expected reward. The action that starts the maximizing sequence is taken, and the process repeats. The problem with this approach is that it is horrifically computationally expensive.

To address this problem, Hutter introduces a general problem-solving algorithm, denoted M^{ϵ}_{p*}. Let p^* be an algorithm that computes $p^*(x)$ for input x. In our case, p^* could be the brute force approach to computing the best next action given the history of past actions and observations. Given p^*, x, and $\epsilon \in (0, \frac{1}{2})$, M^{ϵ}_{p*} provably computes $p^*(x)$ in time that is asymptotically only a factor of $1+\epsilon$ slower than the *fastest* algorithm for computing p^*, despite the fact that this fastest algorithm is unknown. The two questions that immediately come to mind are: How does M^{ϵ}_{p*} work? And where's the catch? We answer both questions below.

Given that p^* and x are inputs to the algorithm, one thing that the algorithm does is start computing $p^*(x)$. In parallel, it enumerates proofs in a formal language (e.g., first-order logic) from shortest to longest, and checks to see if any of these proofs establish that there is some other algorithm p with running time t that computes the same thing as p^*. Each time such a proof is found, the pair (p,t) gets added to a list. A third parallel thread runs through the items in this list and, for each (p,t) pair, computes $t(x)$, which is the amount of time it will take p to compute $p(x)$. If $t(x)$ is smaller than the remaining time required by p^* to compute $p^*(x)$, then the computation of the first thread is stopped, $p*$ is replaced by p, the first thread is restarted, this time computing $p(x)$, and the entire process continues. The fraction of "CPU time" devoted to searching for proofs of the existence of other algorithms for computing p^* is ϵ, as is the fraction devoted to computing $t(x)$ for all (p,t) pairs in the list. That leaves a fraction of $1-2\epsilon$ for computing $p^*(x)$ using the fastest algorithm found so far.

By construction, $M^{\epsilon}_{p*}(x)$ computes the same function as $p^*(x)$, and Theorem 7.1 of Hutter's book establishes that this algorithm is slower than the fastest algorithm for computing $p^*(x)$ by a modest multiplicative factor of $1+\epsilon$. The problem, as Hutter points out, is that there is a lower order term, an *additive* constant, in the run-time complexity of $M^{\epsilon}_{p*}(x)$ that is exponential in the length of the proof that p' computes the same thing as p^*, where p' is the fastest such algorithm. Despite the fact that Theorem 7.2 asserts that "the most efficient program for computing some function f is also among the shortest programs provably computing f", unless the proofs involved in establishing the running time and correctness of the most efficient algorithm are not more than a few bits long, M^{ϵ}_{p*} is completely impractical.

Though M^{ϵ}_{p*} may be impractical, it is computable. Recall that this is not the case with ξ, which is Hutter's approximation to the true probability distribution governing the behavior of the environment and is required to find an optimal policy. The final major contribution of the book is AIXI$_{tl}$, a version of the full AIXI model (i.e., ξ and M^{ϵ}_{p*}) that is computable. The assumption required to make AIXI$_{tl}$ computable is that we, the AI community, are seeking a program that runs on a computer (a real, physical computer with bounded memory) and makes intelligent decisions in a reasonable amount of time. Suppose your computer has l bits of memory, and you think that this intelligent program should react to changes in the environment in no more than t units of time. AIXI$_{tl}$ enumerates all programs for choosing actions based on observations that are no more than l bits long, require no more than t time steps to choose an action, and for which there exist proofs that bound the future rewards the program will obtain. The best such program is allowed to specify the next action to take. This makes AIXI$_{tl}$ computable because programs that do not halt can be terminated after t time steps. However, as with M^{ϵ}_{p*}, there is a large factor hidden inside the asymptotic analysis. In this case, it is a multiplicative factor of 2^l.

3. Conclusion

Many science fiction novels are based on the premise that an extraordinarily powerful computer is built that becomes intelligent. Such novels remind us of a cartoon we saw years ago in which the characters are pouring over a list of three items. The first says "corner the market on athletic socks", the third says "world domination", and they're trying to figure out step 2. Ray Kurzweil has been making lists like that for years: (1) let Moore's law work, (3) Artificial Intelligence. But what is step 2? Just because really fast hardware exists, it is not at all apparent what algorithm should be run to endow that hardware with intelligence. We view Hutter's contribution as providing an initial, formal idea about what step 2 might be.

It would be missing the point entirely to dismiss Hutter's work because the algorithms have exponential computational complexity. He envisioned a world in which computational resources are not an issue and both asked and answered the following two questions: How would I formally specify what it means to be universally and optimally intelligent? What algorithm(s) can be used to implement a universally optimal intelligent system? Just as Solomonoff's work on universal induction prompted decades of work by others, including computable and practical approximations to the universal prior (e.g., minimum message length [9], minimum description length [3], universal similarity metrics [5], and context-tree weighting methods [10]), we expect the same to be true of Hutter's work on universal artificial intelligence.

References

[1] R. Brooks, Personal communication, 2001.
[2] D. Hume, Enquiry Concerning Human Understanding, 1748.
[3] J. Rissanen, Modeling by shortest data description, Automatica 14 (1978) 465–471.
[4] D. McDermott, How intelligent is Deep Blue? ftp://ftp.cs.yale.edu/pub/mcdermott/papers/deepblue.txt, Expanded version of article that first appeared in New York Times, May 14, 1997.
[5] P.M.B. Vitanyi, R. Cilibrasi, Clustering by compression, IEEE Transactions on Information Theory 51 (4) (2005) 1523–1545.
[6] S.J. Russell, P. Norvig, Artificial Intelligence: A Modern Approach, Pearson Education, 2003.
[7] R.J. Solomonoff, A formal theory of inductive inference—part i, Information and Control 7 (1964) 1–22.
[8] R.J. Solomonoff, Complexity-based induction systems, IEEE Transactions on Information Theory 24 (5) (1978) 422–432.
[9] C.S. Wallace, D.M. Boulton, An information measure for classification, Computer Journal 11 (2) (1968) 185–194.
[10] F.M.J. Willems, Y.M. Shtarkov, T.J. Tjalkens, The context-tree weighting method: basic properties, IEEE Transactions on Information Theory (1995) 653–664.
[11] D.H. Wolpert, The supervised learning no-free-lunch theorems, in: Proceedings of the 6th Online World Conference on Soft Computing in Industrial Applications, 2001.
[12] D.H. Wolpert, W.G. Macready, No free lunch theorems for optimization, IEEE Transactions on Evolutionary Computation 1 (1997) 67–82.