

Transcript

Artificial Intelligence as a Precise Art

Artificial General Intelligence Research Institute AGI Workshop

November 1, 2006

Eliezer S. Yudkowsky

[Intro.] Good morning. My name is Eliezer Yudkowsky and I am a Bayesian. The title of this talk is "AI as a Precise Art", and as a Bayesian, I obviously don't think that intelligence consists of carrying out logical deductions on perfect knowledge of the exact state of the environment. But just because you're uncertain about the environment, you shouldn't be afraid to use precise methods to deal with your own uncertainty. I'd like to begin by presenting an object lesson in how *not* to deal with uncertainty.

[Cards.] First, an experiment by Amos Tversky and Ward Edwards in 1966. The subjects were shown a succession of cards, each card either red or blue. 70% of the cards were blue, and 30% red, and the experimenters had randomized the sequence. What Tversky and Edwards discovered was that most subjects, asked to guess the color of each succeeding card, would guess blue around 70% of the time, and red about 30% - as if the subjects thought they could predict the sequence, which was in fact randomized. Even when the subjects were paid a nickel for each correct guess, they still guessed blue an average of only 76% of the time. A better strategy would have been to guess blue every time. If you guess blue all the time, you'll get a nickel 70% of the time, and over a thousand trials you'll earn seven hundred nickels. If you guess at the same frequency as the cards, then 70% of the time you'll guess blue and get a nickel 70% of those times, and 30% of the time you'll guess red and get a nickel 30% of those times, in which case your expected payoff over a thousand trials is five hundred and eighty nickels.

Robyn Dawes, commenting on this experiment, said: "Despite feedback through a thousand trials, subjects cannot bring themselves to believe that the situation is one in which they *cannot* predict." I would reply that the subjects didn't know that the card sequence was randomized; it was rational for them to keep looking for patterns. That's what science is all about, looking for order. On the other hand, even if the subjects thought they noticed a pattern, they didn't need to pay to test their hypotheses. They

could have thought, "If pattern XYZ is true, the next card will be red," and then went on betting on blue until some hypothesis had actually been confirmed. I would suggest that subjects simply didn't think of the strategy of betting on blue every time. It's a counterintuitive betting strategy because it doesn't resemble a typical sequence of cards. If you had perfect knowledge of the cards, then your optimal betting strategy would exactly resemble the cards. But, under conditions of uncertainty, your optimal betting pattern may look startlingly unlike a typical sequence of cards.

A noisy environment does not imply a noisy strategy for dealing with the environment. Or to put it another way, a random key does not fit a random lock just because they are "both random".

[Vinge/chess.] Vernor Vinge once suggested that we would be unable to predict the actions of any sufficiently smart Artificial Intelligence, or any entity smarter than human, on the grounds that if you can predict the exact action of an entity smarter than you are, you are necessarily at least that smart yourself because you can just do whatever a superintelligence would do in your shoes.

Let's try applying this logic to the chess-playing program Deep Blue. You cannot build Deep Blue by programming in a good chess move for every possible chess position; first of all, there are too many chess positions, and second, the resulting program would play no better than you did. If Deep Blue's programmers had known exactly where Deep Blue would move in any particular position, they would necessarily have been at least that good at chess themselves; they could have just moved wherever Deep Blue would move. Now imagine that the programmers had said to themselves: "We want an AI that can transcend our own chess-playing abilities. If we *knew* what Deep Blue's move would be, it couldn't possibly play any better than us. So we'll use a random move generator, and then we won't know where Deep Blue will move. Problem solved!" And yet instead of using a random move generator, the programmers felt obligated to spend a huge amount of time and effort crafting a complex program that would output moves that were predictably so good the programmers couldn't predict them. As Marcello Herreshoff put it, "We never run any computer program unless we know *something* about the output, and we *don't* know the actual output."

[Gilovich.] Here the point is that an unknown key does not fit an unknown lock. If you don't know a superior move, and you don't understand your own

AI program, these two ignorances do not cancel out. This is an important point to keep in mind because, as Thomas Gilovich observes, conclusions people don't want to believe are held to higher standards than conclusions people want to believe. If someone doesn't want to believe something, they ask whether the evidence *compels* them to believe; if someone wants to believe something, they ask whether the evidence *allows* them to accept the conclusion. If Deep Blue's programmers hadn't understood their own program, and hadn't known which chess move would beat Kasparov, they might have convinced themselves that somehow things would magically work out, and no one could have *proven* to them that they were wrong. There are times when people can be very reluctant to relinquish their ignorance for exactly that reason - think of any creationist trying to claim that evolution is just a theory.

[Jaynes.] Neither the uncertainty of the environment, nor the programmers' own uncertainty as to the best or satisficing action, imply that an AI's internal workings need to be mysterious. E. T. Jaynes, the father of maximum entropy methods, once observed that if we are ignorant about a phenomenon, this is a fact about us, not a fact about the phenomenon itself. A blank map does not correspond to a blank territory. Confusion exists in the mind, not in reality. Jaynes labeled this the "Mind Projection Fallacy" - the error of treating our own cognitive states as if they were properties of external objects. The mysteriousness of an AI method cannot be responsible for the AI working well, because our uncertainty about the AI is a property of us, not a property of the AI.

Why is this important to emphasize? Consider this quote from the eminent nineteenth-century physicist, Lord Kelvin. The big mystery in the nineteenth century was what we would call biology, that is, the extraordinary, inexplicable difference between things that are alive and things that aren't. Lord Kelvin said:

[Kelvin.] "The influence of animal or vegetable life on matter is infinitely beyond the range of any scientific inquiry hitherto entered on. Its power of directing the motions of moving particles, in the demonstrated daily miracle of our human free-will, and in the growth of generation after generation of plants from a single seed, are infinitely different from any possible result of the fortuitous concurrence of atoms... Modern biologists were coming once more to the acceptance of something and that was a vital principle."

If you consider this quote, it looks like Lord Kelvin got a tremendous emotional kick out of the mysteriousness of biology. *Infinitely beyond the range of any scientific inquiry!* Not just a *little* beyond the range of science, but *infinitely* beyond! When you get that much satisfaction from mysteriousness, you aren't likely to look kindly on any darned *answers* that come along. Something that seems sufficiently mysterious creates an aesthetic experience, and this aesthetic experience is very dangerous because you can become attached to it. Lord Kelvin saw a mysterious question and thought that it had a mysterious answer. But there are no phenomena that are mysterious in themselves. A blank map does not correspond to a blank territory.

Lord Kelvin's mistake is closely related to the mistake of guessing a mixed sequence of blue and red cards. The observed nature of life seemed very mysterious, so the vitalists hypothesized a cause, the *elan vital*, which was equally sacred and mysterious. A mysterious effect with a mundane cause is just as counterintuitive as a mixed pattern of cards giving rise to an optimal betting strategy of all-blue.

When things are unknown we think they are more beautiful, more powerful, because of that, and this is dangerous. The only reason to become excited about not knowing something is that you intend to solve the puzzle in the immediate future. The glory of glorious mystery is to be destroyed, after which it ceases to be mystery. So I find it disturbing when people, for example, praise the power of genetic programming because they don't know how the resulting programs work, or praise the power of neural networks because they don't know how the network analyzes the data. Or when someone says that the unpredictability of how intelligence emerges in the brain is the key to human creativity. Your ignorance of a phenomenon is a fact about you, not a fact about the phenomenon, so your inability to explain how a process works cannot make it work better.

I have an ulterior motive for saying all this, which is that I need precise understanding of a class of problems where neural nets and genetic programming will both automatically fail because they are not certain to succeed.

[Good.] In 1965, I. J. Good hypothesized an effect he named the "intelligence explosion": If an AI is sufficiently smart, said Good, it can rewrite its own source code to make itself smarter; and then, having become

smarter, rewrite its own source code again, becoming in smarter, and so on ad infinitum. I. J. Good only talked about AI, but in principle the concept of an intelligence explosion generalizes further: for example, humans augmented by direct brain-computer interfaces, using their improved intelligence to design better brain-computer interfaces. In any case, I. J. Good therefore predicted that once an AI became sufficiently smart, a positive feedback cycle would rapidly take it to what Good called "ultraintelligence". To me, Good's intelligence explosion scenario has a great deal of intuitive plausibility. And I definitely think that reflectivity and self-modification is one of the great keys to AI. Thus, I am very interested in the question of how a mind can remain stable while rewriting its own source code.

[Trans1] An analogy: Take a single transistor in your computer's CPU, and ask, given that the transistor is operating today, what is the chance that it will still be functioning tomorrow? Many faults could destroy the transistor; the heatsink could fail and fry the chip, lightning could strike the power line, someone might throw the computer out the window. That doesn't happen every day, but we can loosely state that it happens more often than once in every three thousand years. So if you look at one lone transistor, nothing else, and ask the probability that it will still function tomorrow, the chance of failure is clearly greater than one in a million. [Trans2] But there are millions of transistors in the chip - say 155 million for a top-end CPU in 2006. So, we conclude, if each lone transistor has a failure probability greater than one in a million, then the chance of the entire chip working throughout the day is infinitesimal. [Trans3] The flaw in this reasoning is that the probability of failure is not conditionally independent between transistors. A heatsink failure will destroy many transistors at the same time, so if I'm told that one transistor is operating correctly, my probability that the neighboring transistors operate correctly goes way up. The potential causes of failure that are conditionally independent for each transistor operation must have probabilities very close to zero. A computer chip needs to perform a mole of transistor operations, error-free, every two weeks or so.

[Trans4] For an AI to remain stable while rewriting its own source code, the cumulative probability of catastrophic error has to be bounded over millions of sequential self-modifications. Imagine that when we get to the point where the millionth version of the source code is designing the millionth-plus-one version, nothing went wrong on any of the previous million code changes. This doesn't tell you that the probability of the whole AI failing

was zero, but it tells you that the independent component in the probability of failure on each round of self-modification was effectively zero.

[Chip1] How can we possibly achieve this? Well, how do modern chip engineers design a machine that has 155 million interdependent parts, and that can't be fixed once it leaves the factory? [Chip2] The glorious thing about formal mathematical proof is that a formal proof of ten billion steps is as reliable as a proof of ten steps. The proof is just as strong as its axioms, no more, no less. This doesn't mean the conclusion of a formal proof is perfectly reliable. Your axioms could be wrong. But it is possible, even in the real world, for a formal proof of ten billion steps to be right.

[Chip3] Human beings can't check a purported proof of ten billion steps because we're too easily bored, too slow, and too likely to miss something. [Chip4] And present-day theorem-proving techniques aren't powerful enough to design a chip and prove it correct; they undergo an exponential explosion in the search space. [Chip5] Human mathematicians can prove theorems much more complex than automated theorem-provers can handle. [Chip6] But human mathematics is informal and unreliable, and sometimes we find a flaw in a previously accepted proof. [Solution.] So the humans choose the lemmas, a complex theorem-prover generates a formal proof, and a simple verifier checks the steps. That's how modern engineers build reliable machinery with 155 million interdependent parts. It requires a synergy of human intelligence and computer algorithms because *currently* neither suffices on its own.

If an AI had a similar combination of abilities - both the ability to avoid an exponential explosion, and the ability to verify a proof with extreme reliability - then it might be able to execute a mole of sequential self-modifications with a bounded cumulative failure probability.

But if you require this, it rules out vast swathes of potential AI techniques that are opaque, or stochastic. You can't rely on genetic programming. You can't rely on trial and error. Success on a thousand trials may argue probabilistically for success on the thousandth-and-first trial if the trials are independent and identically distributed, but success on a thousand trials does not imply success on a million trials. Modern theorem-proving techniques do not begin to address the kind of thought that needs to happen, but whatever method we do use will need to work deterministically rather than

stochastically. Usually when I say this, someone replies that we need chaotic minds to handle a chaotic world, hence the first half of this talk.

[Det1] The inside of a chip is a nearly deterministic environment; we ought to be able to succeed with nearly deterministic probability in how the AI rewrites itself, even if we can only succeed probabilistically in the external environment. [Det2] The AI wouldn't prove that its next version would succeed in the real world, but it would prove something about how its code tried to succeed in the real world. If you imagine that the AI currently wants to help little old ladies across the street, the AI wouldn't try to prove that its future self would deterministically succeed in helping the lady across the street with probability 1, but the AI would prove that its future self would do its best to help little old ladies across the street rather than trying to steer them into oncoming trucks. [Det3] The current AI wouldn't know the exact move its future self makes, because the AI's future self is intended carry out a more effective search for good moves, but the current AI would understand the criterion that generated the move.

[Formal] That's an informal argument. I'd like to formalize it, but I've been having trouble doing so. Current decision theory revolves around expected utility maximization, and that formalism can't handle self-reference. You can use expected utility to evaluate the expected utility of an action, or evaluate the expected utility of source code that chooses between actions. But the current formalism has no way to wrap all the way around - no way to modify the source code that does the modifying. The algorithm breaks down on an infinite recursion. The problem is not representable within the system. [Formal2] And yet humans think about themselves all the time; we have no difficulty using the word "I" in a sentence. I can think about what it would be like to modify my own neural circuitry, and my brain doesn't crash. How are humans handling the self-embedding? The significance of this problem is not that the classical algorithm is too inefficient for the real world, as is often the case with theoretical Bayes. Rather, we don't know how to solve this problem even given infinite computing power. We don't have a deep understanding of the structure of the problem; we don't know what kind of work needs to be performed.

[Reflective] I expect that anyone here can think of at least three ad-hoc solutions for getting an AI to modify its own source code. But is an ad hoc solution going to last through a million sequential self-modifications? More importantly, there is something here we don't understand. We have a very

deep understanding of how the Bayesian rules for probability theory arise; we know literally dozens of different constraint sets that will yield the same set of rules for probability theory. Similarly for generating the axioms of expected utility maximization. Jaynes, master of maximum entropy, thought very deeply about the nature of ignorance. Judea Pearl's book on Causality represents an extraordinarily deep understanding of conditional independence, its relation to human perceptions of causality, and why that makes Bayesian networks work well on some kinds of problems. What we need is this kind of deep understanding for reflectivity and self-modification. Designing AI will only become a precise art when we understand how to make an AI design itself. That is my current project, to develop a reflective decision theory. I have not solved this problem but I am working on it, and I am very interested in any suggestions you might have, and especially if you can think of any references I should read. Thank you.