AI Alignment

AISFP 2017

September 13, 2017

Abstract

Notes from the AI summer fellows program 2017.

Contents

1 Cartesian and naturalized agents

1. Consider a robot thinking about a video game: what inputs would cause me to win?

   It thinks about what happens if it puts in inputs A and B; which input would give good outputs? It does something like argmax.

2. Now consider a different world. In this world the robot imagines itself in this world. The game involves robots thinking about things like robots.

In the first setting, the robot thinks: Here's one thing I can do, and what it does to the world. It has copies of the world inside it.

In the second setting: You can't fit a whole copy of the world inside the robot's brain because the robot's brain is in the world. There are no "functions." We don't have a function the robot

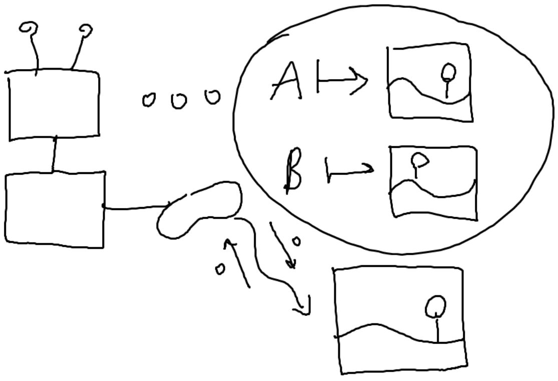

Figure 1: Cartesian agent

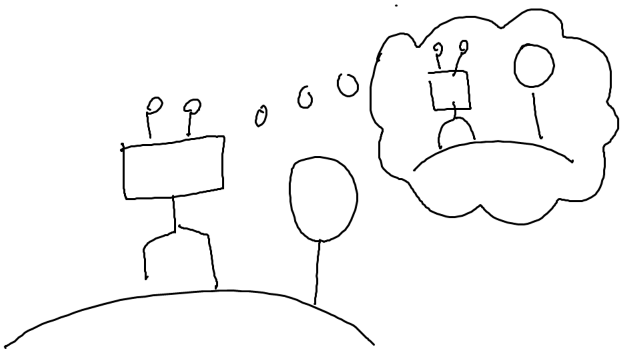

Figure 2: Naturalized agent

knows or a well-defined function the robot can learn about. There's nothing that feels like a "base" function.

In the second world, there could be other things that reason like the robot.

The first setting is a **Cartesian agent**—"plug into" the world. The second is a **naturalized agent**.

Question: what are naturalized agents like?

(You can introduce some complications in the Cartesian case by requiring the robot to be smaller than the world.)

There could be a thing in the video game that is a copy of the robot: the robot affects its input-output channel but also something else in the video game.

1.1 Discussion

1. Q: In practice don't fully simulate the world/other agents. What about a partial simulation? But some robot which did more can exploit you?

   The mindset is: We're focusing on creating Euclidean geometry, not do the engineering.

2. Only thing that controls is input/output. Contents of agent's head and thought process. You can't split it into input/output channel.

3. Imagine I just thought of the world as neurons firing. But isn't that isomorphic to knowing about myself? The thought itself is an action.

   "You're optimizing the thing that is doing the optimizing."

4. The field of game theory has part of "naturalized agency," making the first picture look more like the second. Other agents are off the same type as you, but you view them as perfect adversaries.

5. Why is this interesting from AI alignment perspective?

   Objection: It's better to stick together already existing tools and approaches/bits of confusion (Nash equilibrium, find more things to collide it with) than create something new from scratch.

6. The I/O (Cartesian) model makes it difficult to think about cases when agent is rewriting its own internals. it doesn't help when the agent starts to grow up.

Here's an outline.

1. Cartesian world: First we look at what we understand about the first picture. That took a lot of time but we understand it well now. This gives us something to compare to.

   What parts of what we understand here get messed up in the second picture, and what keeps working?

   Then we do math.

2. Fixed point: Has most useful tools to think about this. Think about how fixed point theory applies to the picture.

3. Philosophy

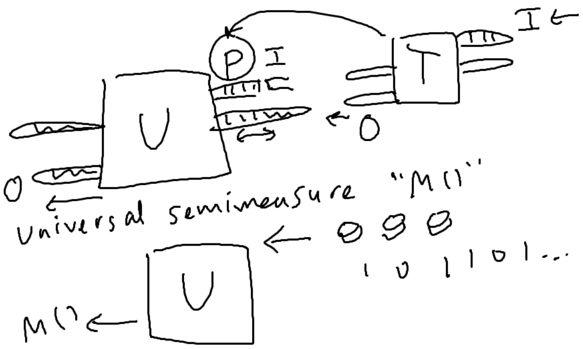

Figure 3: Solomonoff induction

2 Cartesian agency

We'll talk about Solomonoff induction and AIXI.

2.1 Solomonoff induction

Solomonoff induction is the theory of learning if you have unbounded computational power.

This was introduced in the original conference on AI in the 1950's. The form that we have today was perfected in the 1990's.

You can get everything by formalizing Occam's razor very hard. We want to believe the simplest theory consistent with all the data, i.e. the simplest computer program generating all the data so far.

We want a Bayesian prior probability that conforms to this.

Definition 2.1. *A **Turing machine** has three tapes.*

1. *an output tape. Once it outputs a bit it can't rewrite it.*
2. *working tape going infinitely in each direction. You can move in either direction, and rewrite as you want.*
3. *input tape (program tape), can't go backwards.*

*A **universal Turing machine** U is such that for any other Turing machine T in this form, there is a program, i.e., a finite string of bits which we can put as prefix to the input to the universal Turing machine U,[1] before the input to T, so that U will give the same output.*

Definition 2.2. *The **universal semimeasure** is a distribution on infinite bitstrings. When we feed in random bits to the universal Turing machine, we get bits out. $M(p)$ is the probability of getting p as prefix to the output.*

[1] Make sure no program is a prefix of another program. The probability of a program is 2^{-l} where l is the length of the encoding.

Halting issues make M uncomputable, but also makes something interesting happen.

Given 0100111, what's the probability that the next bit is 0/1? They don't sum to 1 because there's a probability that the machine doesn't output anything further.

Solomonoff thought this was a problem. The Solomonoff normalization is as follows.

Definition 2.3 (Solomonoff measure)**.** *Conditioned on a next bit being outputted, what is the probability that it is 0/1?*

Note this is different from conditioning on strings that give infinite output.[2]

The distribution has the following nice property: If l is length of shortest program that gives everything so far, then the probability of the output is at least 2^{-l}. This gives a lower bound on probability of specific observation.

Doing science is about predicting sensory observations, trying to figure out what the input bits might be that emitted the particular output (scientific observations).

Bayes loss quantifies prediction error in sensory observations. The loss is $-\lg(p')$ where p' is the probability you assign to the event that happens (ex. bit that is output).

Any amount of Bayes loss is because the universal measure assigned probability $2^{-l} < 1$ to the correct program.

When you do Bayesian update, you cut out all programs which don't give the correct next bit. You must be increasing the probability of the correct program by $\frac{1}{p'}$. (Renormalize good programs to sum to 1.) I can only increase it so much before it's 1.

I.e., Bayes loss is at most $-\lg\left(\frac{1}{2^l}\right) = l$.

This is nice but also dissatisfying: we assume computable environment. It would be nice to deal with stochastic worlds, randomness in lives. Maybe we'll never learn true equations of physics or use them to fully predict output.

Can Solomonoff induction predict good stochastic models of the world? Yes.

We can emulate stochastic Turing machines. Consider a Turing machine taking in input and using randomness to give output. Use the rest of the coin flips on input tape of the UTM as randomness. (You lose probability mass every time, but not more than you would lose representing a probabilistic hypothesis.)

You get finite loss compared to whatever other machine learning algorithm that works in that environment. (Total loss can be infinite if there is infinite randomness.)

2.2 AIXI

We define a universal (reinforcement learning) agent that uses the universal semimeasure or Solomonoff induction.

The universal Turing machine has a new tape that is the action tape. The agent gives the actions which the environment takes in.

Some number of bits forms a time step. R is a function of these bits and outputs a reward.

Have a finite horizon which you treat as the end of time. Make a tree of possible actions back to the current time. Maximize reward.

We can do things to get rid of the horizon function: use discounted reward and take the limit forwards. Exponential discounting is temporally consistent.

[2] For the Solomonoff measure there's the odd thing of "jumping" to a different model if a model stops giving output at a particular step.

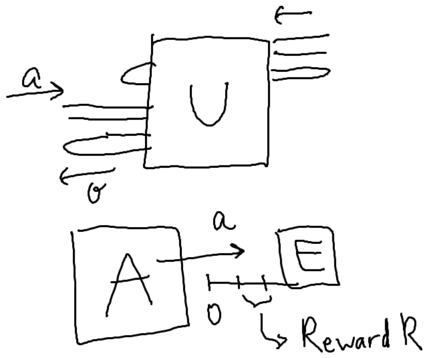

Figure 4: AIXI

In what sense is this optimal? You don't necessarily get to the optimal policy.

It's doing best thing according to universal distribution. You can modify AIXI: do some exploration—ex. open a door to find out what happens. If you do exploration in the right way, you can converge to the right optimal policy. But if there may be traps in the environment—opening the door may lead to infinite hell. "You're doing all the things you should be doing in hell." This is the sense that you can converge to the optimal policy.

Do we want Bayes optimal, optimal thing that falls in traps,...? This is unclear.

3 Advice

1. Have high (self-)standards for your understanding.
   - Original, not novel: "novel" means first time anyone has the idea. "Original" means you can regenerate them, explain them, translate them.
   - Counter-arguments: Run the counterargument train on yourself over and over again.
2. Nimbleness (Notice when you're going down same paths and do something else)
   - Boggle
   - Pre/post-formal distinction:
     (a) Pre-formal means you don't do any proofs. Reason using intuition and algorithms but can't give rigorous statements of theorem.
     (b) Formal: Crystal concepts, concepts and definitions that slot into each other, clear semantics, write formal proofs.
     (c) Post-formal: interact with underlying math reality but still reason in intuitive way, understand the stuff behind the symbols.
   - "Raw thoughts" as epistemic status tag.
3. Attend to the central mysteries, like naturalization. Even as you're doing math, think in the back of your head of agents reasoning; match up math to something that might fit into an agent.

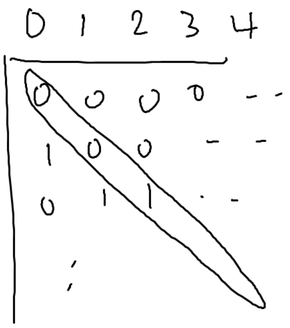

Figure 5: Diagonalization for $\mathbb{N} \twoheadrightarrow P(\mathbb{N}) = 2^{\mathbb{N}}$

4. Courage.

4 Diagonalization

We have a reflection property and fixed point property. In many cases there is a simple relationship.

Definition 4.1. *The* ***reflection*** *property: g is a surjection of a set onto a set of functions whose domain is the same set.*

$$g : S \twoheadrightarrow T^S \tag{1}$$

Definition 4.2. *The* ***fixed point*** *property: For*

$$f : T \to T, \tag{2}$$

$x \in T$ with $f(x) = x$.

Roughly, if you have the reflection property than every function has a fixed point.

Sometimes we use this in the positive direction; sometimes we use the contrapositive.

Fixed point sounds more mathematical, and reflection sounds more like agent foundations.

T is like states of robot. For each state, do something about that.

Think of T^S as a set of functions from S to T (ex. computable functions, continuous functions, etc.).

4.1 Examples

We have a general proof that generalizes the proof that $\neg\exists \mathbb{N} \twoheadrightarrow P(\mathbb{N}) = 2^{\mathbb{N}}$.

To show that statement, consider

$$h : x \mapsto f(g(x)(x))$$

where $f(0) = 1$ and $f(1) = 0$. Suppose by way of contradiction there is $n \in \mathbb{N}$ such that $n \in \mathbb{N}$, $g(n) = h$. Then

$$g(n)(n) = h(n) = f(g(n)(n)), \tag{3}$$

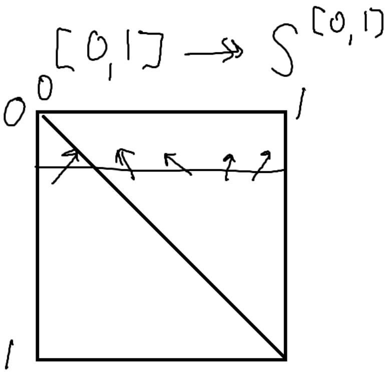

Figure 6: Diagonalization for $[0,1] \twoheadrightarrow \mathbb{S}^{[0,1]}$

so $g(n)(n)$ is a fixed point, contradicting definition of f. The proof is even in the positive direction, it (constructively) gives a fixed point. (Picking the n is less constructive.)

We can also look at topological examples. If we have a map $[0,1] \twoheadrightarrow \mathbb{S}^{[0,1]}$ where $\mathbb{S}$ is the circle (just the boundary).

The sides are the interval. Each "row" is a line of directions. Can we have a map from the square to functions $\mathbb{S}^{[0,1]}$ that lists out all possible maps? No, we can look at the diagonal. This is an "abstract" diagonal.

Currying relates functions of multiple arguments and functions of 1 argument valued in function space. $\widetilde{g}(x)(y)$ is the curried version of $g(x,y)$. $g : \mathbb{N} \to P(\mathbb{N}) = 2^{\mathbb{N}}$ corresponds to $\widetilde{g} : \mathbb{N} \times \mathbb{N} \to \{0,1\}$.

More generally, let $f : T \to T$ and $g : S \to T^S$. For all x, consider the function $S \to T$ defined by

$$s \mapsto f(g(s,s)).$$

The fixed point comes from the element of the set that codes for the function. If there is no fixed point, then the map is not surjective.

Lawvere's fixed point theorem works for all cartesian closed categories (roughly, there is notion of composition of functions and products). All the problems have the same kind of proof, and are corollaries of Lawvere's fixed point theorem.

Quines use the same idea: call function with code of function as argument. There's a version you can do in computability-land, fixed point is not halting. Modify the construction to not have that problem; this tells you how to make quines.

4.2 Lambda calculus

There are also positive uses of Lawvere's fixed point theorem.

Lambda calculus is a minimalistic programming language. Everything is a function, functions are notated literally as lambda terms. For example $(\lambda\, x.\, x)$ is the identity function, and

$$(\lambda\, x.\, x)(\lambda\, x\, y.\, y\, x) = (\lambda\, x\, y.\, y\, x).$$

We can add more datatypes to give better examples. Let's say we also have arithmetic. Then

$$\begin{aligned}(\lambda\, f\, x.\, f\,(f\,x))(\lambda\, y.\, y+3)(5) &= (\lambda\, y.\, y+3)((\lambda\, y.\, y+3)(5)) && (4)\\ &= (\lambda\, y.\, y+3)(8) = 11 && (5)\\ (\lambda\, x.\lambda\, y.\lambda\, z.\, z\, x)\, 7\, (\lambda\, a.\, a+a)\, (\lambda\, b.\, b-5) &= (\lambda\, y.\lambda\, z.\, z\, 7)\, (\lambda\, a.\, a+a)\, (\lambda\, b.\, b-5) && (6)\\ &= (\lambda\, z.\, z\, 7)\, (\lambda\, b.\, b-5) && (7)\\ &= (\lambda\, b.\, b-5)\, 7 = 2 && (8)\\ (\lambda\, x.\lambda\, y.\lambda\, z.\, y\,(z\,x))\,(\lambda\, a.\, 2\cdot a)\,(\lambda\, b.\, b+3)\, 10 = 23. & && (9)\end{aligned}$$

Given $f : \Lambda \to \Lambda$, I want a lambda term x that evaluates to $f\,x$:

$$f \equiv f\,x.$$

This is asking for a fixed point property.

The functions $\Lambda \to \Lambda$ are exactly the lambda terms,

$$\Lambda = \Lambda^{\Lambda}$$

so $f \in \Lambda$. (This can be made precise by domain theory.)

Here we have a bijection $\Lambda \twoheadrightarrow \Lambda^{\Lambda}$, lambda terms just are the right sort of function. g is implicit. Construct x by the diagonal construction.

The diagonal map is $y \mapsto y\,y$ which corresponds to $(\lambda\, y.\, y\,y)$, and the function is $y \mapsto f\,(\lambda\, y.\, y\,y)$ which corresponds to $\lambda\, y.f\,(\lambda\, y.\, y\,y)$. We get

$$(\lambda\, y.f\,(\lambda\, y.\, y\,y))\,(\lambda\, y.f\,(\lambda\, y.\, y\,y)) = f\ ((\lambda\, y.f\,(\lambda\, y.\, y\,y))\,(\lambda\, y.f\,(\lambda\, y.\, y\,y))) \tag{10}$$

This is the Y-combinator. You can use it to implement recursion.

$$Y = \lambda\, f.\,(\lambda\, y.\, f(y\,y))(\lambda\, y.\, f\,(y\,y)). \tag{11}$$

(Note: 2 lambda terms are the same if they can be reduced to the same thing. However, this equivalence is not computable.)

4.3 Applications to logic

I'll discuss applications to logic and to Agent Foundations.

I'll do logic with as little logic as possible. Let L be a language like the language of Peano arithmetic. Let L_n be formulas with n free variables.

These are sentences in L_0:

$$2+2=3 \tag{12}$$
$$\forall x.\, \exists y.\, y > x. \tag{13}$$

These are formulas in L_1:

$$2+n=3 \tag{14}$$
$$\forall x.\, x+n=x. \tag{15}$$

Here x, y are quantified variables and n is a free variable.

$2+2=3$ is a sentence and $\ulcorner 2+2=3 \urcorner$ is code for a sentence (a number).

One application is the diagonal lemma.

Lemma 4.3 (Diagonal lemma)**.** *For all $\varphi \in L_1$, there is $\psi \in L_0$ such that*

$$\vdash \varphi(\lceil \psi \rceil) \leftrightarrow \varphi.$$

"$\vdash$" means provable, $\vdash 2+2=4$, $\nvdash 2+2=3$.

Tarski's undefinability theorem: is there a formula $T \in L_1$ that codes for a true sentence? Suppose there is

$$T(\lceil \psi \rceil) \leftrightarrow \psi.$$

By the Diagonal Lemma, since $\neg T(n) \in L_1$, we can find ψ such that

$$\vdash \psi \leftrightarrow \neg T(\lceil \psi \rceil).$$

4.3.1 Löb's theorem

A second application of the diagonal lemma is Löb's Theorem. Given ψ, let $\Box \lceil \psi \rceil$ be the statement that ψ is provable. This is definable. (There exists a number encoding a proof of it.)

Under which conditions is

$$\vdash \Box \lceil \psi \rceil \to \psi?$$

The box is a formula in our language (that takes in number as input), and the turnstile is meta-mathematically saying that what's to the right can be proved.

This is true when ψ is provable. Also for self-fulfilling proofs. (If agent can prove it turns left, it turns left.) Are there other sentences for which this is provable? Löb's Theorem says no.

Theorem 4.4 (Löb's Theorem)**.**

$$\vdash \Box \lceil \psi \rceil \to \psi \implies \vdash \psi.$$

Suppose an agent is such that if the agent finds a proof that it goes left, it goes left. Then Löb's Theorem says it goes left.

Löb's sentence is constructed using the diagonal lemma.

In the context of an agent, this has weird consequences. We might want it to believe soundness about itself: believe things it's proven, i.e., assert anything it's proven is true. If it assigns probabilities, consider the sentences it assigns probability 1 to. It can self-referentially form the sentence

$$\mathbb{P}(\phi) = 1 \to \phi. \tag{16}$$

This is soundness applied to belief rather than provability.

Löb's Theorem only uses simple facts about provability such as

$$\vdash \Box \lceil \psi \rceil \to \Box \lceil \Box \lceil \psi \rceil \rceil.$$

In the belief case, you can recover many properties of provability. If agent assigns probability 1 to $\mathbb{P}(\phi) = 1 \to \phi$, it has to already believe (assign probability 1) to ϕ.

We can think about the agent as wanting to plan for its future self. It's a desirable property that the agent at time 0 will think it will form correct judgments at time 1. Then the agent at time 1 has to be doing proofs in a different theory than at time 0. If agent at time 1 is doing proofs in PA and so is the agent at time 0, then this property can't hold.

$$\vdash_{ZFC} \Box_{PA} \lceil \phi \rceil \to \phi.$$

Reduction in power is not a satisfying way of solving the problem.

There is a bounded variant of Löb's theorem (up to proofs of certain length).

What if you're always in ZFC, trusting subcomponent of PA?

You can have partial self-modification as long as you can prove to trusted core that modification is OK. But you can't convince the trusted core to modify itself.

Coherence and reflection conditions on probability distribution.

Definition 4.5. $\mathbb{P}$ *is* ***coherent*** *if*

1. *If* $\vdash \perp \varphi'$ *then* $\mathbb{P}(\varphi) = 1$*.*

2. *If* $\vdash \varphi \to \psi$ *then* $\mathbb{P}(\varphi) \leq \mathbb{P}(\psi)$*.*

3. $\mathbb{P}(\varphi \wedge \psi) + \mathbb{P}(\varphi \vee \psi) = \mathbb{P}(\varphi) + \Phi(\psi)$*.*

$\mathbb{P}$ *has* ***reflection*** *if*

$$\mathbb{P}(\varphi) < a \textit{ iff } \mathbb{P}(\mathbb{P}(\varphi) < a) = 1.$$

$\mathbb{P}$ can't be coherent and reflective at the same time.

4.3.2 Löbian handshake

There is 1 positive thing we can do with Löb's Theorem that lets us implement something we would want to implement: Löbian handshakes to implement coordination in certain game-theoretic situations such as Prisoner's dilemma.

If the robot proves that it will turn left, it will turn left (assuming it is provable in the same system).

Consider 2 agents A, B, playing Prisoner's Dilemma, each cooperates if it can prove the other agent cooperates.

$$A = \text{coop} \leftrightarrow \Box \ulcorner B = \text{coop} \urcorner \tag{17}$$

$$B = \text{coop} \leftrightarrow \Box \ulcorner A = \text{coop} \urcorner . \tag{18}$$

We have $A = \text{coop} \wedge B = \text{coop}$ because

$$\vdash \Box \ulcorner A = \text{coop} \wedge B = \text{coop} \urcorner \to A = \text{coop} \wedge B = \text{coop} \tag{19}$$

$$\vdash A = \text{coop} \wedge B = \text{coop}. \tag{20}$$

They are looking for the cooperation handshake. Decide what handshake to look for, define what looking for means...

1. If you replace coop with defect, then both defect.

2. If A coops only if it proves B coops, and B defects only if it proves A defects. Neither can prove what the other does, so A defects and B cooperates.