AI Alignment

AISFP 2017

September 14, 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.

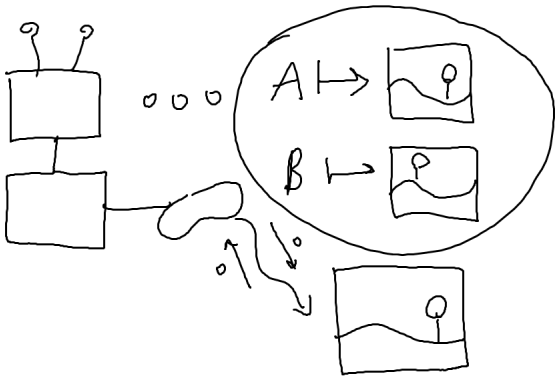


Figure 1: Cartesian agent

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.

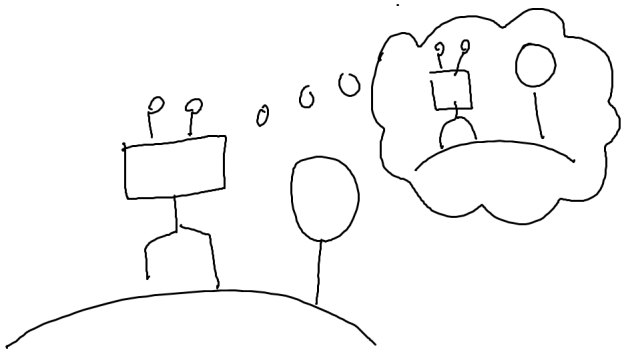


Figure 2: Naturalized agent

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 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

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.*

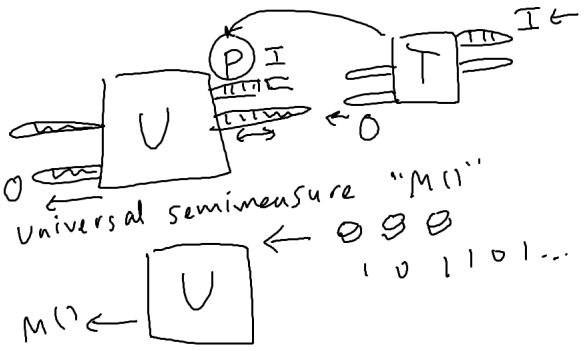


Figure 3: Solomonoff induction

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.

[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.

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.

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

[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.

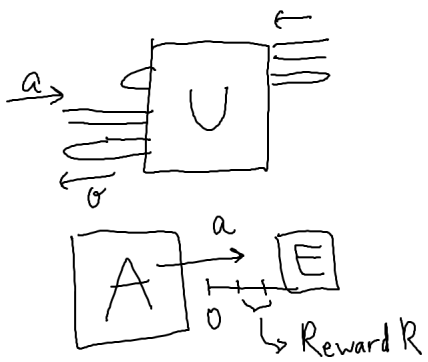


Figure 4: AIXI

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.
4. Courage.

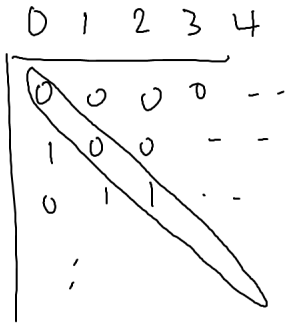


Figure 5: Diagonalization for $\mathbb{N} \twoheadrightarrow P(\mathbb{N}) = 2^{\mathbb{N}}$

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

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

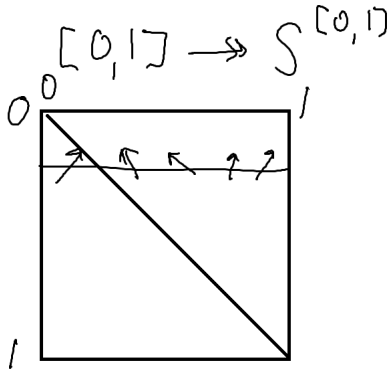


Figure 6: Diagonalization for $[0,1] \twoheadrightarrow \mathbb{S}^{[0,1]}$

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

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

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

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

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)**.**

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

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

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

This is soundness applied to belief rather than provability.

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

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

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

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

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

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

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

$\mathbb{P}$ 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.

5 Game theory

5.1 Nash equilibria

A prisoner's dilemma has the payoff matrix

	C	D
C	2,2	0,3
D	3,0	1,1

No matter what the other player does, I'm better off defecting. It's a dominating strategy to defect. It's a pure Nash equilibrium for both players to defect.

Definition 5.1. (A, B) *is a* ***Nash equilibrium*** *if no player has an incentive to change their actions,*

$$u_1(A, B) \geq u_1(A', B) \qquad \forall A' \tag{21}$$
$$u_2(A, B) \geq u_2(A, B') \qquad \forall B'. \tag{22}$$

There are games for which there doesn't exist a pure Nash equilibrium, for example, matching pennies.

Row player wins if they put pennies the same way and column player wins if they put pennies different ways.

	H	T
H	1,-1	-1,1
T	-1,1	1,-1

First we'll suppose the other player's action is fixed; how does my action control the utility?

1. Suppose $P(p_2 = H) < \frac{1}{2}$. Then I want to pick tails.
2. Suppose $P(p_2 = H) > \frac{1}{2}$. Then I want to pick heads.
3. If $P(p_2 = H) = \frac{1}{2}$, I am indifferent. (E.g. I could randomize.)

The intuition is what you should do is randomized. We can formalize this with **mixed Nash equilibrium**. We have the same defining conditions but the strategies are probabilities.

There is a unique mixed Nash equilibrium for matching pennies.

1. Player 1 plays $\frac{1}{2}H + \frac{1}{2}T$.
2. Player 2 plays $\frac{1}{2}H + \frac{1}{2}T$.

Note this is intentioned with the non-naturalized, Cartesian picture, and it's not clear what the naturalized picture is.

The world cannot simultaneously pick the best thing in response to what you're doing without breaking. We're thinking of this as a 2-player game, responding to what they expect other player to do. This is already bringing in some complexities (but not all) from naturalization. We still

have a well-defined interaction with the world but there's a recursive thing that can go on with the agents.

The world can't optimize to take advantage of you but other agents can. This is a problem if we can't identify agents in the environment; the game theory versus players and the environment is different.

Our outline is the following.

1. Prove existence of Nash equilibria.
2. Philosophy
3. Reflective oracles

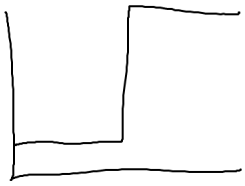
Logical induction gives better philosophical grounding than reflective oracles, but we'll discuss this later.

Kakutani's fixed point theorem is stated in terms of set-valued functions.

Theorem 5.2 (Kakutani's fixed point theorem)**.** *Let S be a nonempty compact convex subset of a Euclidean space.*

Let $f : S \to 2^S$ be a set-valued function (multi-valued function) with a closed graph $\{(s,t) : t = f(s)\}$ pointwise nonempty and convex. (Each $f(s)$ is a nonempty convex set.)

For example, an infinitely steep S-curve is such a multivalued function.



We apply this to $S = S_1 \times S_2$, the total strategy space. Here S_i is the space of mixed strategies over actions available player i, $S_i = \Delta(A_i)$, the simplex of probability distributions over the actions.

For example, when there are 2 actions, $S = [0,1]^2$ is a square.

Then there exists $s \in S$ such that $s \in f(s)$.

Something which is not the same as Kakutani functions but which is useful for intuition pumps is the limit of a sequence of continuous functions (e.g. become steeper and steeper). It also has a fixed point property.

Consider a n-player game, $S = S_1 \times \cdots \times S_n$. Define the best response function $f : S \to \mathcal{P}(S)$ by

$$f(s_1, \ldots, s_n) = \{s_1 : \forall s_1' \in S_1, \quad u_1(s_1, \ldots, s_n) \geq u_1(s_1', \ldots, s_n)\} \times \cdots \subseteq \mathcal{P}(S). \tag{23}$$

Kakutani's Theorem gives the existence of

$$(s_1^*, \ldots, s_n^*) \in f(s_1^*, \ldots, s_n^*).$$

Then no one has an incentive to deviate so this is a Nash equilibrium.

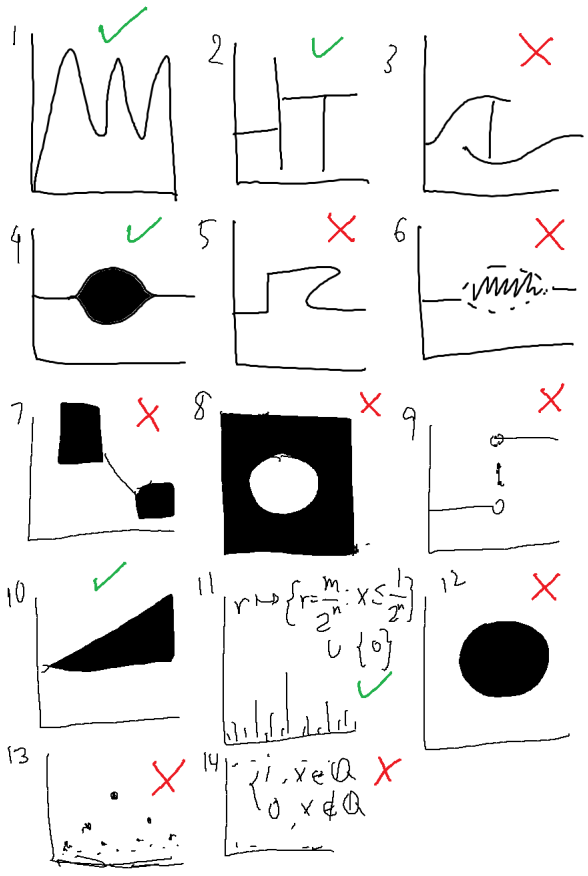


Figure 7: Functions which satisfy the conditions in the Kakutani fixed point theorem

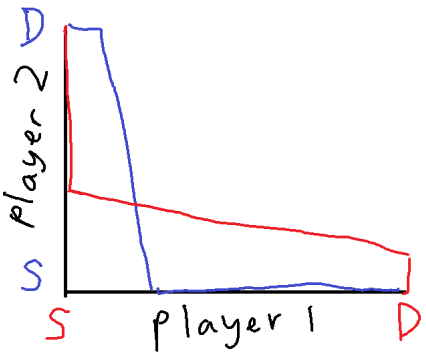


Figure 8: Game of chicken

For example, the game of chicken: we drive towards each other. A person who turns aside (swerve) is chicken and lose; if both people dare, they crash into each other. S is swerve and D is dare.

	S	D
S	0,0	0,1
D	1,0	-10,-10

When player 1 swerves, player 2 definitely wants to dare, when player 1 dares, player 1 definitely wants to swerve.

The 2 functions are individually Kakutani functions, put them together to get $S_1 \times S_2 \to \mathcal{P}(S_1 \times S_2)$.

There are 3 different Nash equilibria where the graphs intersect, 2 pure ones. In the mixed one, each dares with small probability.

5.2 Naturalization

We used Brouwer to do game theory (Brouwer and Kakutani are basically the same thing). Further applications:

1. We use Brouwer to address coherence and reflection: you can't have coherent probabilities and reflection because of L'ob's Theorem, one of the problems Lawvere caused. We used Brouwer to overcome it by assigning infinitesimal probabilities to things.

   Tarski's Theorem: You can't write down what it means for something to be true because then you could diagonalize against yourself: $\neg T(\lceil \phi \rceil) \leftrightarrow \phi$. Instead of having truth values being true or false, we can have continuous truth values. If not is $1 - \bullet$, then we can make the truth value $\frac{1}{2}$... we can use Brouwer to have a consistent way of saying whether or not things are true. We get reflection within any infinitesimal bound, one direction but not the other.

2. We can solve the halting problem by having an oracle that gives the answer. Once you've added, you want to ask when do machines with that oracle halt? You can continuously add

and have a hierarchy. You can add an oracle to the robot, but it's reasoning about things with a oracle, which breaks it.

We want to give something like a halting oracle—reflective oracle—it doesn't just tell you whether normal machine halt, but whether machines with a reflective oracle halt. It does this probabilistically. You ask it questions of the form, does this halt with probability $> \frac{1}{2}$. Answering randomly is like having a Kakutani-type thing.

3. The last thing we use Brouwer for is an agent reasoning about its future self, the Löbstacle. Using logical induction we can model this pretty well; have things that trust themselves almost completely. They manage it without diagonalizing against themselves by using Brouwer.

A lot of problems we had before in Lawvere world we are good in Brouwer world. But in Brouwer we can't come up with a good way to choose the fixed point.

If I were to move my function around, I can bring it back to where it started, and if you move the fixed point continuously you would end up somewhere else.

There are these fixed point god things that choose where you go.

In game theory you have a whole bunch of Nash equilibrium.

It would be nice to have 1 rather than 2 fundamental truths about the universe (Lawvere, Brouwer). I'm curious to whether you can connect them.

We showed we can't have a surjective function $X \twoheadrightarrow (X \to \mathbb{S})$ because there is a map $\mathbb{S} \to \mathbb{S}$ without fixed points.

Is there $X \twoheadrightarrow (X \to [0,1])$? Then you can prove Brouwer using Lawvere. I know what the proof would look like if I can find X...

Another reason I would like this: have model of agents reasoning about other agents. Agents react to other agents' policies.

1. Imagine X is a space of codes (a topological space) a program can have.
2. $X \to [0,1] \to [0,1]^X$ is a space of policies. I look at opponent's code and cooperate with some probability.
3. $X \to (X \to [0,1])$ is interpreter that takes in code and returns policy. Code implements policy $X \to [0,1]$ when you run it.

This seems to connect to fundamental secrets of the universe.

6 Decision theory

I'll work in a space where there's not a distinction between CDT and EDT, or we're using EDT.

We have assignments of probabilities P_n (or expectations $\mathbb{E}_n$) to logical sentences. We build decision theory out of this.

Consider an agent A_n, choosing between action 1 or 0.

$$A_n = \begin{cases} 1, & \text{if } \mathbb{E}_n(U_n|A_m = 1) > \mathbb{E}_n(U_n|A_n = 0) \\ 0, & \text{otherwise.} \end{cases} \tag{24}$$

For logical inductors, for many properties, in the limit, as we make the parameter large enough, the property holds. We would like for A_n to choose the better action in the limit.

For this talk, assume all utilities are bounded. Suppose $U_n = A_n$.

Given all the things logical inductors do well, you might think it might take action 1 in the limit.

Logical inductor learns by feedback. If it takes action 1 all the time, it can never really learn what happens if it takes action 0. It could take action 1 all the time.

Alternatively, it could take action 0 all the time, believe if it takes action 0 it gets 0 reward, and believe if it takes action 1 it gets -10^9 reward.

This is very much like evidential decision theory. The problem is that if you take a conditional on something with probability 0, bad things happen.

You can get this phenomena in all kinds of things. When you condition on something approaching 0, things can be bad.

This is a problem for EDT. For CDT we give it a big structure: this is the node that's you, for each action this is what happens. I understand how to optimize that, using argmax.

In our picture, if I didn't take action A, it's hard to say what happens if I did take action A.

(There may be agents that search for proofs. Isn't there a short proof that $A_n = 1$ gives $U_n = 1$? But there may also be a proof that $A_n = 1$ gives $U_n = -10^9$.)

Note the problem can only happen all the way—if the agent chooses 1 half the time, then it will see what happens in both cases. A (bad) way to deal with this is ε-exploration. Do random thing with probability ε, and learn what things happen.

$$A_n = \begin{cases} 1 & \text{if } \mathbb{P}_n(A_n = 1) < \varepsilon \\ 0 & \text{if } \mathbb{P}_n(A_n = 0) < \varepsilon \\ 1 & \text{if } \mathbb{E}_n(U_n | A_n = 1) > \mathbb{E}_n(U_n | A_n = 0) \\ 0 & \text{otherwise} \end{cases} \tag{25}$$

If you know you're not going to take the action, take the action. This is different from randomizing. If the agent is almost certain won't take action, will take action just to see what happens.

What happens is if it's almost certain it will take an action, it shifts the probability down to $1 - \varepsilon$.

This is like the liar's sentence (this sentence is true with probability $< \frac{1}{2}$—randomize when probability is around $\frac{1}{2}$). The probability would converge to $\frac{1}{2}$ and wiggle so it's above/below it half the time.

This algorithm is unsafe: When you take actions, you don't just lose the local game—What happens if you self-modify into a chicken?

Ignoring this, it works, because when it takes conditional probabilities, it takes them bounded away from 0.

Now consider

$$U_n = 10\mathbb{P}_n(A_n = 0) + A_n. \tag{26}$$

This is playing a prisoner's dilemma against an agent of similar power that thinks "I will coop with probability I think you cooperate."

You want it to believe it's going to choose 0 and play 1. But you can't get both!

I claim that the agent defined by (25) defects all except ε of the time. Suppose it cooperates most of the time. The times it randomly defected, its opponent doesn't predict it, so it gets reward!

When I explore and defect, it starts paying me less. At some point I get suspicious... But if you try cooperating you get even less again...

The logical inductor already knows $\mathbb{P}_n(A_n = 0)$. So when it does expected value calculations, it can treat it as a constant.

Suppose you're playing Prisoner's Dilemma against someone with similar prediction skill to you and knows you about as well as you know yourself. They're going to predict whether or not you cooperate.

Imagine you didn't know the fact, $U_n = 10\mathbb{P}_{f(n)}(A_n = 0) + A_n$ where $f(n)$ is a fast-growing function. Now pretend the predictor is really good. It's going to cooperate.

6.1 Updatelessness

The standard example: I flip a coin and it comes up heads. You trust me and I'm a good predictor of you. I decided I would give you \$100 if it comes up tails but only if I predicted you gave me \$10 if it come up heads.

Now I'm asking you to give me \$10. Do you give me \$10?

It's important that you actually give me \$10 in this situation. If yesterday you knew you wouldn't, you want to self-modify to give me \$10. Then you're not reflectively consistent: you want to modify yourself into something consistent.

I claim that the thing going on in (26) is similar.

The way we solve this is by going to the beginning of time. Rewind the tape, you didn't tell me this yet. In my prior I have some prior probability that the coin comes up heads. I work on that prior probability. I choose the policy (if it comes up heads I give \$10, if it comes up tails I get you \$100).

I don't update on the coin flip. I keep original probabilites on coin flip.

Updatelessness: the coin came up heads but there's this other multiverse where the coin came up tails and I want to help that version.

Updatelessness is good but it's set up in a Bayesian framework.

A prior at the beginning of time is very big, if it has all the logical statements... We don't know how to combine the idea of updatelessness with computation. We can only state it in Bayes's world.

What if instead of "coin is H", it's "millionth digit of π is odd"? Suppose Omega is not adversarial (in choosing the millionth digit). Then if you don't do the same thing, you're not reflectively consistent!

Logical updatelessness is not understood.

7 Honest induction problem

This is also known as Goodhart's law, or the rocket problem.

1. Rocket problem/Goodhart's law
2. Logical induction as robustness solution.

3. Examples/intuitions

We ask a robot to build rockets. The robot goes off and thinks about rockets. The robot comes up with plans for how a rocket should look. It returns a rocket plan to us.

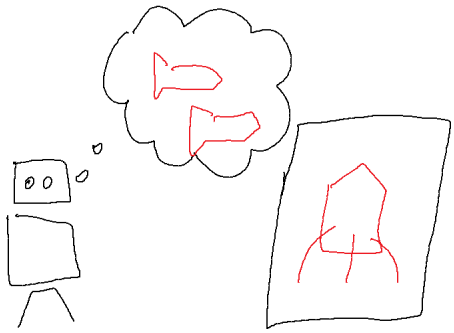


Figure 9: The rocket problem

We look over the plan and evidence that went into its design, and evaluate how good the rocket would be, for example, how good it would be at getting to the moon. There are many reasons this could be hard.

Let E_{me}, E_{robot} be the evidence I and the robot have gathered. We have posteriors

$$\mathbb{P}(\cdot|E_{\text{me}}) \quad \mathbb{P}(\cdot|E_{\text{robot}}).$$

If we were both Bayesians, we can ask the robot to give all its evidence to me, and get

$$\mathbb{P}(\cdot|E_{\text{me}} \wedge E_{\text{robot}}).$$

I would be at an epistemic advantage compared to me or the robot before. We can't do this with logical inductors.

The most important sense in which logical inductors aren't Bayesian is that we can't point to the evidence.

Can we look into a Solomonoff inductor or neural network and ask why does it believe what it believes?

Goodhart's law says: When a measure is used as a target, it ceases to be a good measure. In normal cases it is correlated with some desired objective. When it is being optimized, it becomes an extreme case and it may stop being correlated with the objective.

A lot of scenarios of AI going wrong are like this, e.g. paperclip optimizer.

We have good results about logical induction. How will a logical inductor actually do in practice?

Consider $n = 10^{10}$. What traders have a lot of wealth on this day, and what are they predicting. Traders that did a good job early on are wealthy. They found ways the market or other traders were being stupid. A trader has a list of other statements and considers what it should bet on. It has a large amount of time to compute various facts that help: what to think about more, should I think about what other traders do, do I look for inconsistencies between different facts on the market... It has computational resources and has to think about how to allocate them.

Personifying the trader, I think some simple relationships should hold; if they are outside bounds, make a trade on it. This doesn't mean these traders will earn a lot of money early on. Traders have initial weight dependent on complexity of program describing trader. It seems this is relatively simple and uses little computation.

I expect things like this exist and get money early on; later on other traders stop throwing money away, the trader starts being more cautious, market becoming more rational over time... Specifying all these things separately seems more complicated than having general-purpose optimizer.

If I ask my logical inductor about statements, I'm asking the trader, what will you tell me about this statement?

If it is optimized to do other things, it would try to control the external world, and the first thing it would do (for instrumental reasons) is to make good predictions. Making good predictions so far is often a proxy for making good predictions in the future. But will it continue to make good predictions? Why should I expect it to continue to make good predictions?

I want to update on what my robot believes about rockets. Then we have this Bayesian intuition that I have a good idea what happens in the future. Since we don't have this Bayesian picture, we can't just update on evidence.

Traders give output, and we have this Goodhart question of whether we can trust the output.

Solomonoff induction is a reasonable prior but has all computations inside of it. Any computation could have something to say about the world.

We want to impute traders as having beliefs. Logical inductor has beliefs, but the traders are just structured to be an algorithm.

7.1 Naturalization as instance of honest induction problem

If I had Solomonoff induction, sequence prediction predicting all the facts about the universe, then maybe the universe is simple, and we can trust the prediction.

That's not a situation that's physically possible to put a predictor into. Instead the predictor has e.g. cameras looking at particular things. I'm interested in photons coming from this tree. This is way more algorithmic information than the universe.

We ask: which hypothesis has short description of this camera feed?

General purpose optimizers making predictions of the world with an incentive can be specified in fewer bits than that. It's fewer bits to follow the camera by being a general-purpose optimizer than having a general theory.