AI Safety

AISFP 2017

September 13, 2017

Abstract

Abstract here.

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.

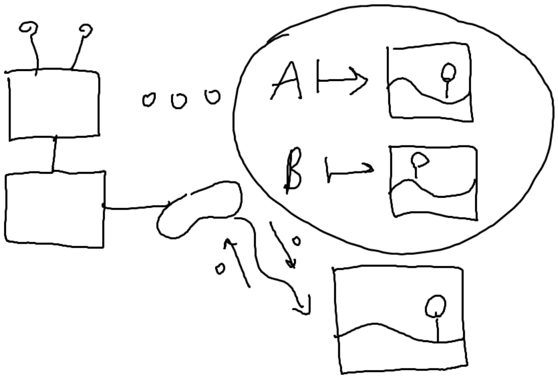

Figure 1: Cartesian agent

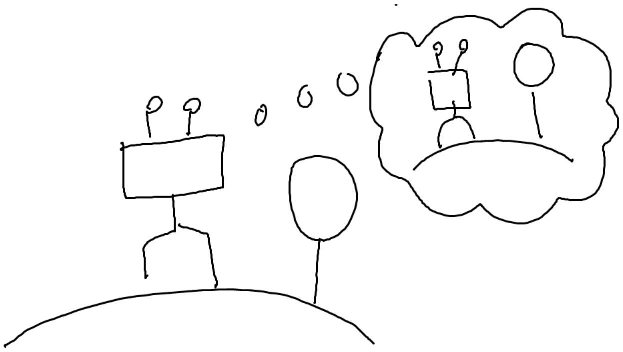

Figure 2: Naturalized agent

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.*

[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.

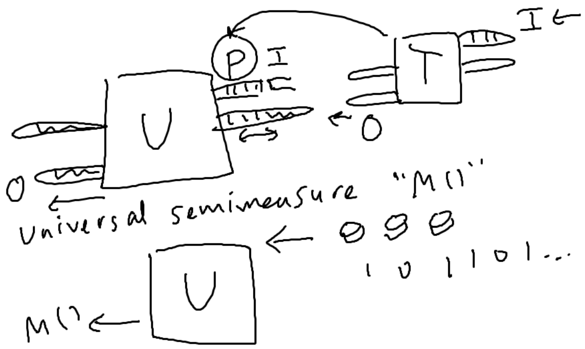

Figure 3: Solomonoff induction

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.*

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.

[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.

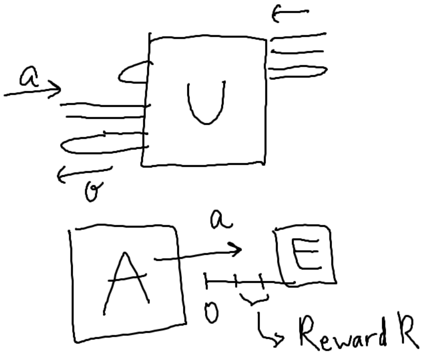

Figure 4: AIXI

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 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.