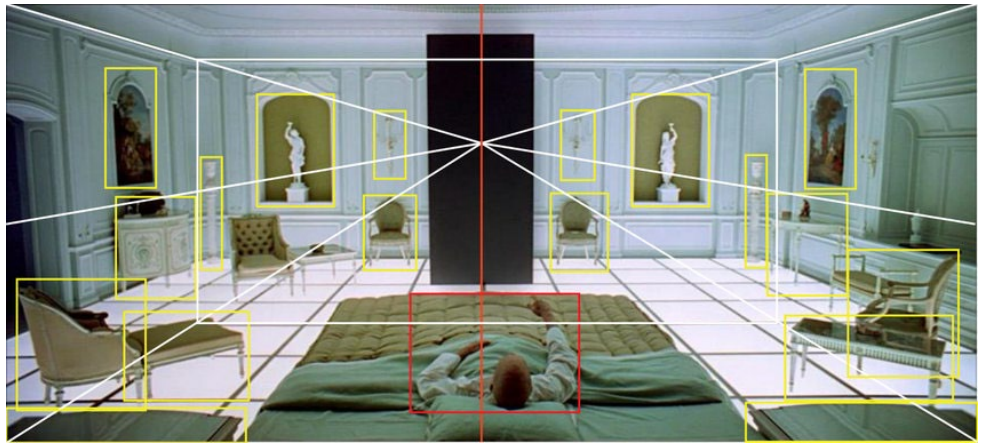
Reward-Punishment Symmetric Universal Intelligence

Samuel Alexander, SEC, samuelallenalexander@gmail.com

Marcus Hutter, DeepMind & ANU



Background (Legg & Hutter 2007)

- Legg/Hutter (2007) distilled the essence of intelligence into a universal form of RL.

- Agent π's *universal intelligence* $\Upsilon(\pi)$ is its average reward over all suitable environments, where each environment μ has weight $2^{-K(\mu)}$ where *K* is Kolmogorov complexity.

- Note *K* depends on the choice of a Universal Turing Machine.

Background (Leike & Hutter 2015)

- Leike/Hutter (2015) noted that $\Upsilon(\pi)$'s dependence on UTM choice is non-trivial. Different UTMs yield radically different universal intelligence measures.

- They posed an open question:

  "What are other desirable properties of a UTM?"

Alexander/Hutter (2021) made progress by considering how intelligence should change if rewards and punishments are swapped.

Preliminaries: Ambient Space

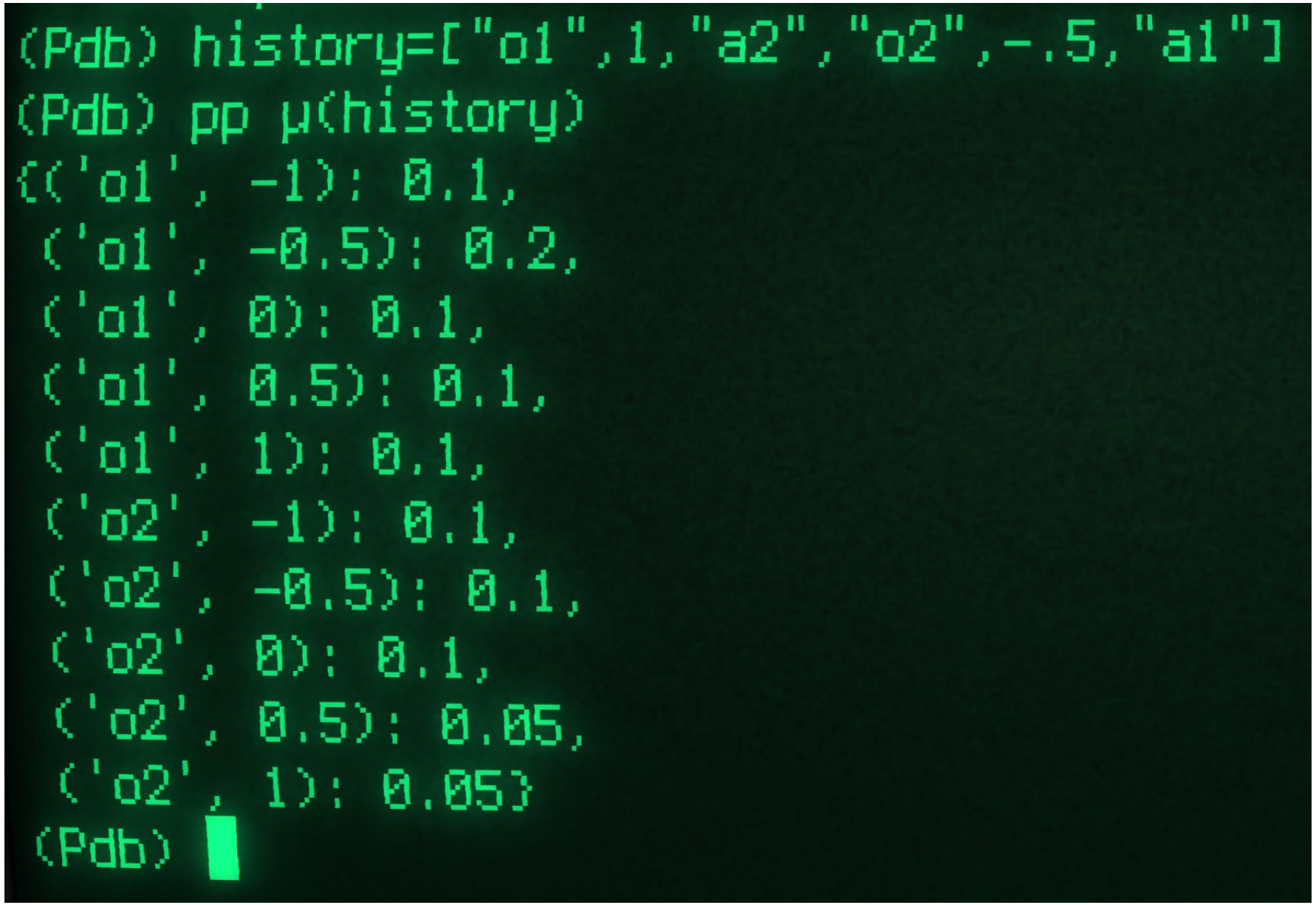
- *A* is a finite nonempty set of *actions*
- *O* is a finite nonempty set of *observations*
- *R* is a finite nonempty set of *rewards*, rational numbers from [-1,1], symmetric about 0 (i.e., *-r* is a valid reward whenever *r* is).

Note: In Legg/Hutter 2007, *R* is only allowed to contain nonnegative rewards.

Definition 1: RL Framework

- Define *(ORA)** and *(ORA)*OR* using regex. E.g., *(ORA)** contains all sequences (obs, reward, action, ..., obs, reward, action).
- An *agent* is a function π assigning rational probability $\pi(a|s)$ for all *a* in *A*, *s* in *(ORA)*OR*.
- An *environment* is a function μ assigning rational probability $\mu(o,r|s)$ for all *o* in *O*, *r* in *R*, *s* in *(ORA)**.
- V_μ^π is the expected total reward π would obtain from μ.

This is a very universal form of RL, like in 17.3 of Sutton & Barto.


(Pdb) history=["o1",1,"a2","o2",-.5,"a1"]
(Pdb) pp μ(history)
{('o1', -1): 0.1,
 ('o1', -0.5): 0.2,
 ('o1', 0): 0.1,
 ('o1', 0.5): 0.1,
 ('o1', 1): 0.1,
 ('o2', -1): 0.1,
 ('o2', -0.5): 0.1,
 ('o2', 0): 0.1,
 ('o2', 0.5): 0.05,
 ('o2', 1): 0.05}
(Pdb)

```python
class PracticalAgent:
    def __init__(self):
        ...
    def act(self, obs):
        ...
        return action_probability_distr
    def train(self, o_prev, a, r, o_next):
        ...

def π(s):  # s in (ORA)*OR
    worker = PracticalAgent()
    while len(s) > 2:
        o_prev, _, a, o_next, r = s[:5]
        worker.train(o_prev, a, r, o_next)
        s = s[3:]

    most_recent_obs = s[0]
    return worker.act(most_recent_obs)

```

Definition 2: Well-behaved environments

Environment μ is *well-behaved* if:

1. μ is computable.
2. For all π, $-1 \leq V_{\mu}^{\pi} \leq 1$.

Note: This is a more universal way to achieve what some authors contrive with tricks like discount factors, etc.

Definition 3: Dual Agents/Environments

- For history s in *(ORA)** or *(ORA)*OR*, $\bar{s}$ is the result of multiplying all rewards by -1.

- For agent π, $\bar{\pi}$ is the agent who confuses rewards and punishments:

$$\bar{\pi}(a|s) = \pi(a|\bar{s})$$

- For environment μ, $\bar{\mu}$ is the environment which switches rewards and punishments:

$$\bar{\mu}(o, r|s) = \mu(\mathrm{o}, -\mathrm{r}|\bar{s})$$

Reward/Punishment Algebra

Lemma 4: $\bar{\bar{x}} = x$ (for any agent, environment, or history x).

Theorem 5: $V_{\bar{\mu}}^{\bar{\pi}} = -V_{\mu}^{\pi}$

Corollary 6: $V_{\bar{\mu}}^{\pi} = -V_{\mu}^{\bar{\pi}}$

Corollary 7: μ is well-behaved iff $\bar{\mu}$ is well-behaved.

Theorem 5 *Suppose μ is an environment and π is an agent. Then*

$$V_{\overline{\mu}}^{\overline{\pi}} = -V_{\mu}^{\pi}$$

(and the left-hand side is defined if and only if the right-hand side is defined).

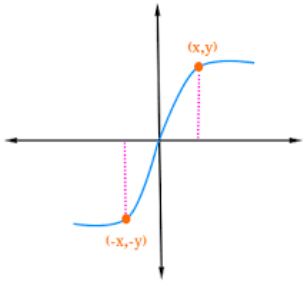
Proof. By Definition 1 part 6, it suffices to show that for each $n \in \mathbb{N}$, $V_{\overline{\mu},n}^{\overline{\pi}} = -V_{\mu,n}^{\pi}$. For that, it suffices to show that for every $s \in ((\mathcal{ORA})^*) \cup ((\mathcal{ORA})^*\mathcal{OR})$, the probability X of generating s using π and μ (as in Definition 1 part 5) equals the probability X' of generating $\overline{s}$ using $\overline{\pi}$ and $\overline{\mu}$. We will show this by induction on the length of s.

Case 1: s is empty. Then $X = X' = 1$.

Case 2: s terminates with an action. Then $s = t \frown a$ for some $t \in (\mathcal{ORA})^*\mathcal{OR}$. Let Y (resp. Y') be the probability of generating t (resp. $\overline{t}$) using π and μ (resp. $\overline{\pi}$ and $\overline{\mu}$). We reason: $X = \pi(a|t)Y = \pi(a|\overline{t})Y = \overline{\pi}(a|\overline{t})Y$ by definition of $\overline{\pi}$. By induction, $Y = Y'$, so $X = \overline{\pi}(a|\overline{t})Y'$, which by definition is X'.

Case 3: s terminates with a reward. Similar to Case 2. □

An Axiom about Symmetry



Suppose $\Upsilon(\pi)$ is the intelligence of π, measured as expected performance averaged (somehow) over all well-behaved environments.

$\bar{\pi}$ uses all π's ingenuity to seek *punishment*, so it seemed natural to suggest as an axiom:

$$\Upsilon(\bar{\pi}) = -\Upsilon(\pi).$$

As further justification, we'll argue this full symmetry axiom is implied by a weaker symmetry assumption.

Justification Step 1: Weak Symmetry

- Assume Υ measures intelligence as average performance.
- Say Υ is *weak symmetric* if: whenever $\Upsilon(\pi) \neq 0$ then $\Upsilon(\pi) \neq \Upsilon(\bar{\pi})$.

Weak symmetry is a reasonable/natural requirement: Say $\Upsilon(\pi)>0$. This should mean π is intelligent: π uses ingenuity to get positive rewards. By def., $\bar{\pi}$ uses that same ingenuity to obtain *punishments*. So it would be strange for $\bar{\pi}$ to get the *exact* same average rewards as π!

Step 2: Weak Symmetry implies Symmetry

Let π be any agent. Assume Υ is weak symmetric and measures intelligence as average performance.

Let ρ be an agent who, at the start of every environment, flips a coin and thereafter plays as π if HEADS, $\bar{\pi}$ if TAILS.

Since Υ measures avg. performance, $\Upsilon(\rho) = \frac{\Upsilon(\pi)+\Upsilon(\bar{\pi})}{2}$.

Define ρ' the same but swap HEADS and TAILS. ρ seems indistinguishable from ρ' so $\Upsilon(\rho) = \Upsilon(\rho')$. Swapping HEADS and TAILS is the same as swapping π and $\bar{\pi}$, thus $\rho' = \bar{\rho}$. Thus $\Upsilon(\rho) = \Upsilon(\bar{\rho})$. By weak symmetry, $\Upsilon(\rho) = 0$.

Thus $\Upsilon(\bar{\pi}) = -\Upsilon(\pi)$!

Toward symmetric Legg-Hutter intelligence

- Legg-Hutter intelligence depends on the choice of a UTM.
- Can we choose the UTM so as to make LH intelligence symmetric?

Actually ... LH intelligence *also* depends on how RL is encoded.

This is usually suppressed.

We need to make it explicit to answer the above question.

Definition 9: RL-Encodings

Legg-Hutter intelligence takes place in a context where computable functions map finite binary strings to finite binary strings.

Definition 9: An *RL-encoding* is a function Π (w/prefix-free range, see next slide) sending RL environment inputs/outputs to binary strings. $We\ write\ \ulcorner x \urcorner\ for\ \sqcap(x).$

A computable function f (taking binary strings to binary strings) *encodes* RL environment μ if:

$$f(\ulcorner s \urcorner) = \ulcorner \mu(\bullet|s) \urcorner$$

Some technical details

In the Legg-Hutter intelligence context, computable functions' domains are *prefix-free* (a set is *prefix-free* if it contains no p,p' such that p is a strict initial segment of p').

So an RL-encoding needs to have prefix-free *range* so it can be composed with computable functions.

Further... Definition 9 (part 2): RL-encoding Π is *suffix-free* if its range never includes p,p' such that p is a strict terminal segment of p'.

Example Prefix-Free Suffix-Free RL-Encoding

Encode histories as strings defining Python arrays (and convert to binary using ASCII):

```
["obs0", 0, "punch", "obs3", 0.5, "kick", "obs2", -1, "kick", "obs1", 0, "punch"]
```

Encode probability distributions on $O \times R$ as strings defining Python dictionaries (and convert to binary using ASCII):

```
{("obs0",0):.5, ("obs0",1):.1, ("obs0",-1):0, ("obs0",0.5):.1, ("obs0",-0.5):.05, ("obs1",0
```

Prefix-free: Code ends with] or }, and] or } occur nowhere else

Suffix-free: Code starts with [or {, and [or { occur nowhere else

Definition 10: Kolmogorov Complexity

Let *U* be a prefix-free UTM (PFUTM), i.e., a UTM with prefix-free domain. Let Π be an RL-encoding.

- For each computable environment μ, the *Kolmogorov Complexity* of μ according to *U* and Π, written $K_U^{\Pi}(\mu)$, is the length of the smallest U-computer program defining a function that encodes μ.

- *U* is Π-*symmetric* if for all μ, $K_U^{\Pi}(\mu) = K_U^{\Pi}(\overline{\mu})$.

Theorem 11: For every suffix-free RL-encoding Π, there is a Π-symmetric PFUTM.

Proof: Let U_0 be some PFUTM. Let U be the PFUTM with U(1X)= U_0(X) and with U(0X) defined as follows:

- If X is U_0 -program "plug history *s* into function *F* to get rational probability distribution $\mu(\bullet|s)$", then instead plug $\bar{s}$ into *F*. If this yields a rational probability distribution m on $O \times R$, then U(0X)=$\overline{m}$, where $\overline{m}(o,r) = m(o,-r)$. Else, U(0X) diverges.

By construction, whenever 0X is a U-code for μ, then 1X is a U-code for $\bar{\mu}$, & vice versa. So U is Π-symmetric. Suffix-freeness is used to show U is prefix-free.

Existence Theorem Illustrated

Assume A={“a”}, O={“o”}, R={1,-1}

U:

```
1def μ(ORA):return {("o",1):.75, ("o",-1):.25}

μ(ORA=["o",1,"a","o",-1,"a"])
```

→

U_0=Python:

```
def μ(ORA):return {("o",1):.75, ("o",-1):.25}

μ(ORA=["o",1,"a","o",-1,"a"])
```

```
0def μ(ORA):return {("o",1):.75, ("o",-1):.25}

μ(ORA=["o",1,"a","o",-1,"a"])
```

→

```
def μ(ORA):return {("o",1):.75, ("o",-1):.25}

def μ_wrapper(ORA):
  m = μ(ORA)
  return {(o,-r):p for (o,r),p in m.items()}

μ_wrapper(ORA=["o",-1,"a","o",1,"a"])
```

Implicit Bias in RL

Our existence proof works by removing bias.

RL researchers have arbitrarily decided “positive good, negative bad”.

It would be just as valid to decide “negative good, positive bad”.

The proof of Theorem 11 can be thought of as constructing a programming language where every program must begin with a bit specifying which of these two conventions the program uses.

Definition 12 *Let W be the set of all well-behaved environments.*

Definition 13 *For every PFUTM U, RL-encoding $\sqcap$, and agent π, the* Legg-Hutter universal intelligence of π given by $U, \sqcap$, *written $\Upsilon_U^{\sqcap}(\pi)$, is*

$$\Upsilon_U^{\sqcap}(\pi) = \sum_{\mu \in W} 2^{-K_U^{\sqcap}(\mu)} V_\mu^\pi.$$

Thm 14: If U is $\sqcap$-symmetric then $\Upsilon^{\sqcap}_{U}(\overline{\pi}) = -\Upsilon^{\sqcap}_{U}(\pi)$.

$$
\begin{aligned}
\Upsilon^{\sqcap}_{U}(\overline{\pi}) &= \sum_{\mu \in W} 2^{-K^{\sqcap}_{U}(\mu)} V^{\overline{\pi}}_{\mu} && \text{(Definition 13)} \\
&= -\sum_{\mu \in W} 2^{-K^{\sqcap}_{U}(\mu)} V^{\pi}_{\overline{\mu}} && \text{(Corollary 6)} \\
&= -\sum_{\mu \in W} 2^{-K^{\sqcap}_{U}(\overline{\mu})} V^{\pi}_{\overline{\mu}} && (U \text{ is } \sqcap\text{-symmetric}) \\
&= -\sum_{\mu \in \overline{W}} 2^{-K^{\sqcap}_{U}(\mu)} V^{\pi}_{\mu} && \text{(Change of variables)} \\
&= -\sum_{\mu \in W} 2^{-K^{\sqcap}_{U}(\mu)} V^{\pi}_{\mu} && (\text{By Corollary 7, } W = \overline{W}) \\
&= -\Upsilon^{\sqcap}_{U}(\pi). && \text{(Definition 13)}
\end{aligned}
$$

Corollary 16 *Let $\sqcap$ be an RL-encoding, let U be a $\sqcap$-symmetric PFUTM and suppose π is an agent which ignores rewards (by which we mean that $\pi(\bullet|s)$ does not depend on the rewards in s). Then $\Upsilon_U^{\sqcap}(\pi) = 0$.*

Proof. The hypothesis implies $\pi = \overline{\pi}$, so by Theorem 14, $\Upsilon_U^{\sqcap}(\pi) = -\Upsilon_U^{\sqcap}(\pi)$. $\square$

Exercise: Permutations

Say a PFUTM is Π-*permutable* if for each permutation P of the action-set *A*, whenever μ' is the environment obtained from μ by permuting actions using P, then K(μ')=K(μ).

By similar reasoning as above, if Π is suffix-free, then Π-permutable PFUTMs exist. In the corresponding Legg-Hutter universal intelligence measure, action permutations preserve agents' intelligence.

Likewise for permutations of the observation-space.

Whether to take absolute values

- If a subject scores 0% on a 1000-question True-False IQ test, are they highly intelligent or highly unintelligent?
- Legg & Hutter measure intelligence purely as average performance: as if to say, the above subject is highly unintelligent. This is contrary to certain everyday intuitions.
- An alternate intelligence measure would average $\left|V_\mu^\pi\right|$ instead of V_μ^π.

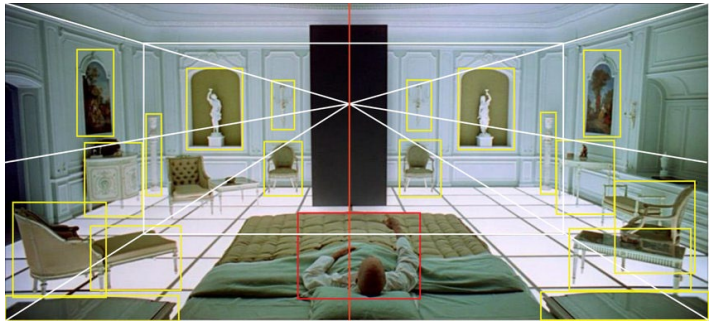
Then "$\Upsilon(\bar{\pi}) = -\Upsilon(\pi)$" would become "$\Upsilon(\bar{\pi}) = \Upsilon(\pi)$".

Our stance on abs values

- Taking abs values or not taking abs values yields two different intelligence measures with different properties.
- Neither is more “valid” than the other (as far as we know). One measures raw average performance, the other measures ability to consistently extremize performance (whether in the good-performance direction or the bad-performance direction).

Arguably, the debate traces back to Plato’s “Lesser Hippias”. Socrates initially seems pro-abs-values, then in a plot-twist he turns his logic against itself (dualizes it?), making him anti-abs-values apparently...

Conclusion



- The *dual* of an agent (resp. environment) is the version of that agent (resp. environment) which swaps rewards/punishments.

- Legg-Hutter intelligence satisfies $\Upsilon(\bar{\pi}) = -\Upsilon(\pi)$ provided the background UTM is symmetric, i.e., that $\mathrm{K}(\bar{\mu}) = \mathrm{K}(\mu)$.

- This symmetry requirement is an example of an intrinsically desirable property of a UTM (aka programming language) in context of RL.

Call for Co-Authors

I'd be interested in working with members of this seminar on follow-up papers. If interested, email me: samuelallenalexander@gmail.com

Some ideas:

- Notions of UTM symmetry in general (not limited to RL context).
- Deeper dive into the "weak symmetry implies symmetry" argument.
- Legg-Hutter intelligence in *almost-symmetric* UTMs (leads naturally to number systems with infinities and infinitesimals).

Supplemental Slides

We write 2^* for the set of finite binary strings. We write $f :\subseteq A \to B$ to indicate that f has codomain B and that f's domain is some subset of A.

Definition 8 *(Prefix-free universal Turing machines)*

1. *A partial computable function* $f :\subseteq 2^* \to 2^*$ *is* prefix-free *if the following requirement holds:* $\forall p, p' \in 2^*$, *if* p *is a strict initial segment of* p', *then* $f(p)$ *and* $f(p')$ *are not both defined.*
2. *A* prefix-free universal Turing machine *(or* PFUTM*) is a prefix-free partial computable function* $U :\subseteq 2^* \to 2^*$ *such that the following condition holds. For every prefix-free partial computable function* $f :\subseteq 2^* \to 2^*$, $\exists y \in 2^*$ *such that* $\forall x \in 2^*$, $f(x) = U(y \frown x)$. *In this case, we say* y *is a* computer program for f in programming language U.

Environments do not have domain $\subseteq 2^*$, and they do not have codomain 2^*. Rather, their domain and codomain are $(\mathcal{ORA})^*$ and the set of $\mathbb{Q}$-valued probability measures on $\mathcal{O} \times \mathcal{R}$, respectively. Thus, in order to talk about their Kolmogorov complexities, one must encode said inputs and outputs. This low-level detail is usually implicit, but we will need (in Theorem 11) to distinguish between different kinds of encodings, so we must make the details explicit.

Definition 9 *By an* RL-encoding *we mean a computable function* $\sqcap : (\mathcal{ORA})^* \cup M \to 2^*$ *(where* M *is the set of* $\mathbb{Q}$*-valued probability-measures on* $\mathcal{O} \times \mathcal{R}$*) such that for all* $x, y \in (\mathcal{ORA})^* \cup M$ *(with* $x \neq y$*),* $\sqcap(x)$ *is not an initial segment of* $\sqcap(y)$*. We say* $\sqcap$ *is* suffix-free *if for all* $x, y \in (\mathcal{ORA})^* \cup M$ *(with* $x \neq y$*),* $\sqcap(x)$ *is not a terminal segment of* $\sqcap(y)$*. We write* $\ulcorner x \urcorner$ *for* $\sqcap(x)$*.*

Definition 10 *(Kolmogorov Complexity) Suppose U is a PFUTM and $\sqcap$ is an RL-encoding.*

1. *For each computable environment μ, the* Kolmogorov complexity of μ given by $U, \sqcap$, *written $K_U^{\sqcap}(\mu)$, is the smallest $n \in \mathbb{N}$ such that there is some computer program of length n, in programming language U, for some function $f :\subseteq 2^* \to 2^*$ such that for all $s \in (\mathcal{ORA})^*$, $f(\ulcorner s \urcorner) = \ulcorner \mu(\bullet|s) \urcorner$ (note this makes sense since the domain of $\sqcap$ in Definition 9 is $(\mathcal{ORA})^* \cup M$).*
2. *We say U is* symmetric in its $\sqcap$-encoded-environment cross-section *(or simply that U is* $\sqcap$-symmetric*) if $K_U^{\sqcap}(\mu) = K_U^{\sqcap}(\overline{\mu})$ for every computable environment μ.*

Theorem 11: For every suffix-free RL-encoding Π, there is a Π-symmetric PFUTM.

Proof: Let U_0 be some PFUTM. Let U be the PFUTM with U(1X)= U_0(X) and with U(0X) defined as follows:

- If X is U_0 -program "plug history *s* into function *F* to get rational probability distribution $\mu(\bullet,\bullet|s)$", then instead plug $\bar{s}$ into *F*. If this yields a rational probability distribution m on $O \times R$, then U(0X)=$\bar{m}$, where $\bar{m}(o,r) = m(o,-r)$. Else, U(0X) diverges.

U is prefix-free by Π-suffix-freeness. By constr., whenever 0X is a U-code for μ, then 1X is a U-code for $\bar{\mu}$, & vice versa. So U is Π-symmetric.

Abs Values History: Plato's "Lesser Hippias"

- Whether to take absolute values is an ancient debate.
- In Plato's "Lesser Hippias", Socrates presents what initially seems like a compelling argument in favor of taking absolute values.

SOCRATES: "Which of the two then is a better runner? He who runs slowly voluntarily, or he who runs slowly involuntarily?" Etc. etc. etc...

Abs Values History: Socrates' Evil Twist

- From what initially seems like a pro-abs-values argument, Socrates uses the same logic to defend the ludicrous position that it's better to be intentionally evil than unintentionally evil.

(An interesting AGI safety question. Is an intentionally evil AGI better or worse than an unintentionally evil one?)

The dialogue ends with poor Hippias hopelessly confused. Better not to take sides on the abs-value question.