

Distribution of Environments in Formal Measures of Intelligence

Bill Hibbard

University of Wisconsin – Madison
SSEC, 1225 W. Dayton St., Madison, WI 53706

Abstract

This paper shows that a constraint on universal Turing machines is necessary for Legg's and Hutter's formal measure of intelligence to be unbiased. It also explores the relation of the No Free Lunch Theorem to formal measures of intelligence.

Introduction

A formal definition of intelligence can provide a well-defined goal to those developing artificial intelligence. Legg's and Hutter's formal measure of the intelligence of agents interacting with environments provides such a definition (Legg and Hutter 2006). Their model includes weighting distributions over time and environments. The point of this paper is to argue that a constraint on the weighting over environments is required for the utility of the intelligence measure.

A Formal Measure of Intelligence

In Legg's and Hutter's measure, based on reinforcement learning, an agent interacts with its environment at a sequence of discrete times, sending action a_i to the environment and receiving observation o_i and reward r_i from the environment at time i . These are members of finite sets A , O and R respectively, where R is a set of rational numbers between 0 and 1. The environment is defined by a probability measure $\mu(o_k r_k | o_1 r_1 a_1 \dots o_{k-1} r_{k-1} a_{k-1})$ and the agent is defined by a probability measure $\pi(a_k | o_1 r_1 a_1 \dots o_{k-1} r_{k-1} a_{k-1})$.

The value of agent π in environment μ is defined by the expected value of rewards:

$$V_\mu^\pi = \mathbf{E}(\sum_{i=1}^{\infty} w_i r_i)$$

where the $w_i \geq 0$ are a sequence of weights for future rewards subject to $\sum_{i=1}^{\infty} w_i = 1$ (Legg and Hutter combined the w_i into the r_i). In reinforcement learning the w_i are often taken to be $(1-\gamma)\gamma^{i-1}$ for some $0 < \gamma < 1$. Note $0 \leq V_\mu^\pi \leq 1$.

The intelligence of agent π is defined by a weighted sum of its values over a set E of computable environments. Environments are computed by programs, finite binary strings, on some prefix universal Turing machine (PUTM) U . The weight for $\mu \in E$ is defined in terms of its Kolmogorov complexity:

$$K(\mu) = \min \{ |p| : U(p) \text{ computes } \mu \}$$

where $|p|$ denotes the length of program p . The intelligence of agent π is:

$$V^\pi = \sum_{\mu \in E} 2^{-K(\mu)} V_\mu^\pi.$$

The value of this expression for V^π is between 0 and 1 because of Kraft's Inequality for PUTMs (Li and Vitányi 1997): $\sum_{\mu \in E} 2^{-K(\mu)} \leq 1$.

Legg and Hutter state that because $K(\mu)$ is independent of the choice of PUTM up to an additive constant that is independent of μ , we can simply pick a PUTM. They do caution that the choice of PUTM can affect the relative intelligence of agents and discuss the possibility of limiting PUTM complexity. But in fact a constraint on PUTMs is necessary to avoid intelligence measures biased toward specific environments:

Proposition 1. Given $\mu \in E$ and $\varepsilon > 0$ there exists a PUTM U_μ such that for all agents π :

$$V_\mu^\pi / 2 \leq V^\pi < V_\mu^\pi / 2 + \varepsilon$$

where V^π is computed using U_μ .

Proof. Fix a PUTM U_0 that computes environments. Given $\mu \in E$ and $\varepsilon > 0$, fix an integer n such that $2^{-n} < \varepsilon$. Then construct a PUTM U_μ that computes μ given the program "1", fails to halt (alternatively, computes μ given a program starting with between 1 and n 0's followed by a 1, and computes $U_0(p)$ given a program of $n+1$ 0's followed by p . Now define K using U_μ . Clearly:

$$2^{-K(\mu)} = 1/2$$

And, applying Kraft's Inequality to U_0 :

$$\sum_{\mu' \neq \mu} 2^{-K(\mu')} \leq 2^{-n} < \varepsilon.$$

So $V^\pi = V_\mu^\pi / 2 + X$ where $X = \sum_{\mu' \neq \mu} 2^{-K(\mu')} V_{\mu'}^\pi$ and $0 \leq X < \varepsilon$. \square

Whatever PUTM is used to compute environments, all but an arbitrarily small ε of an agent's intelligence is determined by its value in a finite number of environments. Lucky choices of actions at early, heavily weighted time steps in simple, heavily weighted environments, may give a less intelligent agent an advantage greater than ε , that a more intelligent agent cannot make up by good choices of actions in very difficult, but lightly weighted environments.

Note that as environment complexity increases, agents will require longer times to learn good actions. Thus, given a distribution of time weights that is constant over all environments, even the best agents will be unable to get any value as environment complexity

increases to infinity. It would make sense for different environments to have different time weight distributions.

Two points for consideration despite their being discouraged by reviewers of this paper:

1. If PUTM programs were answers (as in Solomonoff Induction, where an agent seeks programs that match observed environment behavior) then weighting short programs more heavily would make sense, since shorter answers are better (according to Occam's razor). But here they are being used as questions and longer programs pose more difficult questions so arguably should be weighted more heavily.

2. The physical universe contains no more than 10^{90} bits (Lloyd 2002) so is a finite state machine (FSM). Hence an intelligence measure based on FSMs is more realistic than one based on Turing machines.

No Free Lunch and a Finite Model

The No-Free-Lunch Theorem (NFLT) tells us that all optimization algorithms have equal performance when averaged over all finite environments (Wolpert and Macready 1997). It is interesting to investigate what relation this result has to intelligence measures that average agent performance over environments.

To define an intelligence measure based on finite environments take the sets A , O and R of actions, observations and rewards as finite and fixed. An environment is defined by a FSM:

$$f: S \times A \rightarrow S \times O \times R$$

where S is a finite set of states. The value of an agent in this environment is the expected value of a weighted sum over a finite sequence of future rewards, with weights summing to 1. The measured intelligence of an agent is a weighted sum of its values in environments whose state set sizes fall in a finite range, weights summing to 1.

This finite model lacks an important hypothesis of the NFLT: that the optimization algorithm never makes the same action more than once. The same result can be achieved by a no repeating state condition (NRSC) on environment FSMs: that they can never repeat the same state. Although this may seem artificial, it applies in the physical universe because of the second law of thermodynamics.

Assuming the NRSC and that all FSMs with the same number of states share the same environment weight and the same sequence of time weights, then all agents have the same measured intelligence, the average reward $(\sum_{r \in R} r) / |R|$ (Hibbard 2008).

Conclusion

According to current physics the universe is a FSM satisfying the NRSC. If we measure agent intelligence using a distribution of FSMs satisfying the NRSC in which all FSMs with the same number of states have the same weight, then all agents have the same measured intelligence. In this environment distribution past behavior of environments provides no information about their future behavior. For a useful measure of

intelligence, environments must be weighted to enable agents to predict the future from the past. This is the idea behind Kolmogorov complexity: to more heavily weight environments that can be generated by short programs since agents can more easily learn their behaviors.

However, Proposition 1 shows that a PUTM must be chosen carefully in an intelligence measure based on Kolmogorov complexity. This suggests a distribution of environments based on program length but less abstract than Kolmogorov complexity. So define a PUTM based on an ordinary programming language.

States and behaviors never repeat in the physical world, but human agents learn to predict future behavior in the world by recognizing current behavior as similar to previously observed behaviors and making predictions based on those previous behaviors. Similarity can be recognized in sequences of values from unstructured sets such as $\{0, 1\}$, but there are more ways to recognize similarity in sequences of values from sets with metric and algebraic structures such as numerical sets. Our physical world is described largely by numerical variables, and the best human efforts to predict behaviors in the physical world use numerical programming languages. So the sets A and O of actions and observations should be defined using numerical values, just as rewards are taken from a numerical set R . Including primitives for numerical operations in environment programs has the effect of skewing the distribution of environments toward similarity with the physical world.

An ordinary numerical programming language is a good candidate basis for a formal measure of intelligence. But the real point of this paper is that distributions over environments pose complex issues for formal intelligence measures. Ultimately our definition of intelligence depends on the intuition we develop from using our minds in the physical world, and the key to a useful formal measure is the way its weighting distribution over environments abstracts from our world.

References

- Hibbard, 2008. <http://www.ssec.wisc.edu/~billh/g/de.pdf>
- Legg, S. and M. Hutter. Proc. A Formal Measure of Machine Intelligence. *15th Annual Machine Learning Conference of Belgium and The Netherlands (Benelearn 2006)*, pages 73-80.
<http://www.idsia.ch/idsiareport/IDSIA-10-06.pdf>
- Li, M. and P. Vitányi, *An Introduction to Kolmogorov Complexity and Its Applications, 2nd ed.*. Springer, New York, 1997. 637 pages.
- Lloyd, S. Computational Capacity of the Universe. *Phys.Rev.Lett.* 88 (2002) 237901.
<http://arxiv.org/abs/quant-ph/0110141>
- Wolpert, D. and W. Macready, No Free Lunch Theorems for Optimization. *IEEE Transactions on Evolutionary Computation* 1, 67. 1997.
<http://ic.arc.nasa.gov/people/dhw/papers/78.pdf>