Measuring Agent Intelligence via Hierarchies of Environments

Bill Hibbard

SSEC, University of Wisconsin-Madison,
1225 W. Dayton St., Madison, WI 53706, USA
test@ssec.wisc.edu

Abstract. Under Legg's and Hutter's formal measure [1], performance in easy environments counts more toward an agent's intelligence than does performance in difficult environments. An alternate measure of intelligence is proposed based on a hierarchy of sets of increasingly difficult environments, in a reinforcement learning framework. An agent's intelligence is measured as the ordinal of the most difficult set of environments it can pass. This measure is defined in both Turing machine and finite state machine models of computing. In the finite model the measure includes the number of time steps required to pass the test.

Introduction

This paper proposes an alternative to Legg's and Hutter's measure of intelligence using a reinforcement learning (RL) model [1]. In a simple reinforcement learning model, an agent interacts with its environment at a sequence of discrete times, sending actions to the environment and receiving observations and rewards (rational numbers between zero and one) from the environment at each time step. The value of an agent in an environment is the expected sum of rewards over all time steps, weighted so that the sum lies between zero and one. The intelligence of the agent is the weighted sum of its value in all computable environments, where the weight of an environment is determined by its Kolmogorov complexity. Essentially, this is the length of the shortest program for a prefix universal Turing machine (PUTM) that computes the environment [2]. These weights are such that an agent's intelligence lies between zero and one. The choice of PUTM must be constrained to avoid bias in this measure [3]. There are at least two ways in which this measure is inconsistent with our intuitions about measuring human intelligence:

1. It gives less credit for environments defined by longer programs even though they are usually more difficult for agents. Given arbitrarily small $\varepsilon > 0$, total credit for all but a finite number of environments is less than ε. That is, total credit for all environments greater than some level C of complexity is less than ε, whereas credit for a single simple environment will be much greater than ε. This is not the way we judge human intelligence.

2. It sums rewards from the first time step, with no time to learn. AIXI always makes optimal actions [4] (as long as it is defined using the same universal Turing machine used to define the measure [3]), but AIXI is not computable. We allow humans time to learn before judging their intelligence.

Hernández-Orallo and Dowe address the first difficulty [5] via a modified version of Legg's and Hutter's measure. However, their modified measure employs a finite number of environments and hence cannot resolve differences between agents above some finite level of intelligence.

Hierarchies of Environments for Measuring Intelligence

Prediction is the essence of intelligence, as Hutter makes clear [4] with his use of Solomonoff's prior [6]. This is the prior probability of sequences based on algorithmic complexity as measured by lengths of PUTM programs that compute the sequences, which can be used to estimate the probabilities of sequence continuations. However, this prior does not account for computing resources. Schmidhuber did this with his speed prior [7], in which the probability of sequences combines algorithmic complexity and computing time. Taken with Legg's work on sequence prediction [8] this suggests measuring intelligence via a game of adversarial sequence prediction [9], in which the agent's adversary has a given amount of computing resources. This is related to Scmidhuber's work on predictability minimization [10], where he defined a general principle for learning based on a set of units that mutually try to avoid prediction by one another.

This paper defines a framework for measuring agent intelligence using the game of adversarial sequence prediction against a hierarchy of increasingly difficult sets of environments. Agent intelligence is measured as the highest level of environments against which it can win the game. In this framework, N is the set of positive integers, $B = \{0, 1\}$ is a binary alphabet, B^* is the set of finite binary sequences (including the empty sequence), and B^∞ is the set of infinite binary sequences. An *evader* e and a *predictor* p are defined as programs for a universal Turing machine that implement total functions $B^* \to B$. A pair e and p interact, where e produces a sequence $x_1\, x_2\, x_3 \ldots \in B^\infty$ according to $x_{n+1} = e(y_1\, y_2\, y_3 \ldots y_n)$ and p produces a sequence $y_1\, y_2\, y_3 \ldots \in B^\infty$ according to $y_{n+1} = p(x_1\, x_2\, x_3 \ldots x_n)$. The predictor p wins round $n+1$ if $y_{n+1} = x_{n+1}$ and the evader e wins if $y_{n+1} \neq x_{n+1}$. We say that the predictor p *learns to predict* the evader e if there exists $k \in N$ such that $\forall n > k$, $y_n = x_n$ and we say the evader e *learns to evade* the predictor p if there exists $k \in N$ such that $\forall n > k$, $y_n \neq x_n$.

Let $t_p(n)$ denote the number of computation steps performed by p before producing y_n and $t_e(n)$ denote the number of computation steps performed by e before producing x_n. Given any computable monotonically increasing function $f: N \to N$, define E_f = the set of evaders e such that $\exists k \in N, \forall n > k.\ t_e(n) < f(n)$ and define P_f = the set of predictors p such that $\exists k \in N, \forall n > k.\ t_p(n) < f(n)$. Then [9] proves the following:

Proposition 1. Given any computable monotonically increasing function $f: N \to N$, there exists a predictor p_f that learns to predict all evaders in E_f and there exists an evader e_f that learns to evade all predictors in P_f.

We can interpret a predictor p as an agent and an evader e as an environment and say the agent p passes at environment e if p learns to predict e. Note that this is a deterministic model of agents and environments. This battle of predictor and evader trying to simulate each other is much like minmax chess algorithms, which themselves are a metaphor for life's competition.

Let $\{g_i: N \to N \mid i \in N\}$ be an enumeration of primitive recursive functions [11], define $h_i(k) = \max\{g_i(j) \mid j \leq k\}$, and define $f(m)$: $N \to N$ by $f(m)(k) = \max\{h_i(k) \mid i \leq m\}$. Then define a hierarchy of sets of environments (evaders) $\{E_{f(m)} \mid m \in N\}$ used in the following definition:

Definition 1. The intelligence of an agent p is measured as the greatest m such that p learns to predict all $e \in E_{f(m)}$ (use $m = 0$ if p cannot satisfy this for $m = 1$).

Proposition 2. In Proposition 1, if f: $N \to N$ is primitive recursive then the computing times of p_f and e_f constructed in the proposition's proof are primitive recursive.

Proof. First note that primitive recursive functions are precisely the functions that can be implemented by loop programs (essentially, these are programs that use ordinary arithmetic and for-loops where the number of iterations is computed before the loop begins) [12]. The proof of Proposition 1 in [9] constructs p_f (and equivalently e_f) by, at time step n, enumerating all universal Turing machine programs of length $\leq n$ and running each for up to $f(n)$ time steps, then doing some simple computations with the results. The enumeration of programs with length $\leq n$ can be done by a loop program and $f(n)$ can be computed by a loop program since it is primitive recursive, so p_f is computed by a loop program. The computing time of any loop program is primitive recursive [12]. □

In order to measure low levels of intelligence, the first enumeration g_i of primitive recursive functions should be ordered to start with functions with small values. For example, $g_i(\mathrm{k}) = i$ for $i \leq 100$, $g_i(\mathrm{k}) = (i - 100) * k$ for $100 < i \leq 200$.

Propositions 1 and 2 imply the following property of the intelligence measure in Definition 1:

Proposition 3. Any agent p whose computing time is bounded by a primitive recursive function must have finite intelligence, and given any integer $n \geq 1$ there is an agent p with intelligence $\geq n$ whose computing time is primitive recursive.

A Hierarchy of Finite State Machines

According to current physics the universe contains only a finite amount of information [13], so finite state machines (FSMs) provide more realistic models than Turing machines.

So we model predictors and evaders as FSMs. An evader e has a state set S_e, an initial state I_e and a mapping $M_e : B \times S_e \to S_e \times B$, and similarly for predictor p, state set S_p, initial state I_p and mapping $M_p : B \times S_p \to S_p \times B$. The timing is such that $(es_{n+1}, x_{n+1}) = M_e(y_n, es_n)$ and $(ps_{n+1}, y_{n+1}) = M_p(x_n, ps_n)$ (with the convention that $x_0 = y_0 = 0$, $es_0 = I_e$, and $ps_0 = I_p$ for the mappings in the initial time step). As in the Turing machine case, the predictor p wins round n+1 if $y_{n+1} = x_{n+1}$ and the evader e wins if $y_{n+1} \neq x_{n+1}$. We say that the predictor p *learns to predict* the evader e if there exists $k \in$

N such that $\forall n > k$, $y_n = x_n$ and we say the evader e *learns to evade* the predictor p if there exists $k \in N$ such that $\forall n > k$, $y_n \neq x_n$.

Define $t(e)$ and $t(p)$ as the number of states in S_e and S_p. Given any $m \in N$, define E_m = the set of evaders e such that $t(e) \leq m$ and define P_m = the set of predictors p such that $t(p) \leq m$. We can prove the following:

Propostition 4. Given any $m \in N$, there exists a predictor p_m that learns to predict all evaders in E_m and there exists an evader e_m that learns to evade all predictors in P_m.

Proof. Construct a predictor p_m as follows:

```
// Initialization
Fix some ordering of $E_m$ and initialize W = $E_m$
For each $e \in W$:
   Initialize $e$'s simulated state s$_e$ = $I_e$
$y_0$ = 0
// Interacting with an evader
For time step $n \geq 1$:
   If $W$ is empty:
      // Interact with an evader not in $E_m$
      Output 0 and input x$_n$
   Else:
      Pick $e$ as the first member of $W$
      ($s$, $x$) = $M_e(y_{n-1}, s_e)$
      $y_n$ = $x$
      // Interact with an evader that may be in $E_m$
      Output $y_n$ and input $x_n$
      For each $e \in W$:
         ($s_e$, $x$) = $M_e(y_{n-1}, s_e)$
         If $x \neq x_n$ remove $e$ from $W$
```

E_m is finite so this predictor p_m is a finite state machine. Assume that p_m interacts with an evader $e \in E_m$. (If W becomes empty, then the algorithm is interacting with an evader that is not a member of E_m.) For each evader e' previous to e in the fixed ordering of E_m set $n_{e'}$ = the time step n, in the interaction between p_m and e, when e' is removed from W. If e' is never removed from W, set $n_{e'} = 0$. Set $k = \max\{n_{e'} \mid e' \text{ previous to } e \text{ in } E_m\}$. Now at each time step $n > k$, each evader e' previous to e in the ordering of E_m is either removed from W or produces output equal to the output of e. Thus p_m correctly predicts e for all time steps $n > k$. That is, p_m learns to predict e.

Now we can construct an evader e_m using the program that implements p_m modified to complement the binary symbols it writes to its output tape. The proof that e_m learns to evade all predictors in P_m is the same as the proof that p_m that learns to predict all evaders in E_m, with the obvious interchange of roles for predictors and evaders. □

As with Proposition 1, interpret a predictor p as an agent and an evader e as an environment, so at time step n action $a_n = y_n$, observation $o_n = x_n$ and reward $r_n = 1$ when $y_n = x_n$ and $r_n = 0$ when $y_{n+1} \neq x_{n+1}$. Furthermore, say the agent p passes at environment e if p learns to predict e. Note that this is a deterministic model of agents and environments.

Then define a hierarchy of sets of environments (evaders) $\{E_m \mid m \in N\}$ used in the following definition:

Definition 2. The intelligence of an agent p is measured as the greatest m such that p learns to predict all $e \in E_m$ (use $m = 0$ if p cannot satisfy this for $m = 1$). If $m > 0$ then since E_m is finite there exists $t \in N$ such that p predicts all $e \in E_m$ past time step t (i.e., $\forall n > t$, $y_n = x_n$ in the interaction between p and all $e \in E_m$). We say this t is the time within which agent p achieves intelligence n. It provides a finer measure of intelligence than m, so we use (m, t) as a detailed measure of p's intelligence. Note that in (m, t) increasing m and decreasing t indicates increasing intelligence.

Propostion 4 implies the following property of the intelligence measure in Definition 2:

Proposition 5. Any FSM-based agent p must have finite intelligence, and given any integer $n \geq 1$ there is a FSM-based agent p with intelligence (m, t) where $m \geq n$.

Discussion and Conclusion

As presented by Hutter, prediction is fundamental to intelligence [4]. This paper has showed how to measure intelligence via prediction ability. The measure for FSM-based agents is universal to all levels of intelligence and, in the Turing machine model, the measure is universal to all levels of intelligence for agents with primitive recursive computing time. Furthermore, the measures are based on long term behavior of agents, giving them time to learn. The measure for FSM-based agents includes a term for the rate at which agents learn. Thus these measures address the two problems discussed in the introduction.

Agents have finite intelligence according to these measures because they can always be defeated by environments that use sufficient computing resources, hence quantity of computing resources is one important component in determining agent intelligence. Goertzel defines an “efficient pragmatic general intelligence” measure that normalizes for quantity of computing resources [14]. This is an interesting idea, but there is utility in a measure of an agent’s ability to succeed in environments regardless of the quantity of computing resources it uses.

Tyler has set up a web site for tournaments among agents playing the Matching Pennies Game, which is mathematically identical with adversarial sequence prediction [15]. He limits the computational resources agents may use.

It would be interesting to investigate an intelligence measure based on Schmidhuber's speed prior. For example, the measure of Legg and Hutter could be modified by replacing Kolmogorov complexity by Levin complexity (essentially, the sum of Kolmogorov complexity and the log of computing time) [16], as used in the speed prior. Alternatively, the measure in Definition 1 could be modified replacing simple computing time by Levin complexity. It would be interesting to investigate other generalizations of the measures in Definitions 1 and 2. We may allow agents to pass their tests by predicting evaders in some proportion α of time steps less than 1.0 (but greater than 0.5). We may also be able to define hierarchies of environments with more than two possible actions and observations.

References

1. Legg, S., Hutter, M.: A Formal Measure of Machine Intelligence. In: 15th Annual Machine Learning Conference of Belgium and The Netherlands (Benelearn 2006), pp. 73-80. Ghent (2006) http://www.idsia.ch/idsiareport/IDSIA-10-06.pdf
2. Li, M., Vitányi, P.: An Introduction to Kolmogorov Complexity and Its Applications, 2nd ed.. Springer, New York (1997)
3. Hibbard, B.: Bias and No Free Lunch in Formal Measures of Intelligence. J. of Artificial General Intelligence. **1**, 54-61 (2009) http://journal.agi-network.org/portals/2/issues/JAGI_1_54-61.pdf
4. Hutter, M.: Universal Artificial Intelligence: Sequential Decisions Based on Algorithmic Probability. Springer, Berlin (2004)
5. Hernández-Orallo, J., Dowe, D.: Measuring universal intelligence: Towards an anytime intelligence test. Artificial Intelligence. **17**, 1508-1539 (2010)
6. Solomonoff, R. J.: A Formal Theory of Inductive Inference: Parts 1 and 2. Information and Control **7**, 1-22 and 224-254 (1964)
7. Schmidhuber, J.: The Speed Prior: A New Simplicity Measure Yielding Near-Optimal Computable Predictions. In: Proc. 15th Annual Conference on Computational Learning Theory (COLT 2002), Lecture Notes in Artificial Intelligence, Springer. 216--228, 2002.
8. Legg, S.: Is there an Elegant Universal Theory of Prediction? Tech. Report No. IDSIA-12-06 (2006) http://www.idsia.ch/idsiareport/IDSIA-12-06.pdf
9. Hibbard, B.: Adversarial Sequence Prediction. In: The First Conference on Artificial General Intelligence (AGI-08). pp. 399-403. IOS Press, Amsterdam (2008) http://www.ssec.wisc.edu/~billh/g/hibbard_agi.pdf
10. Schmidhuber, J.: Learning Factorial Codes by Predictability Minimization. Neural Computation **4**(6), 863-879 (1992)
11. Liu, S.-C.: An enumeration of the primitive recursive functions without repetition. Tohoku Math J. **12**, 400-402 (1960)
12. Meyer, A.R., Ritchie, D.M.: The complexity of loop programs. Proc. of the ACM National Meetings. pp. 465-469. ACM, New York (1967)
13. Lloyd, S.: Computational Capacity of the Universe. Phys.Rev.Lett. **88**, 237901 (2002) http://arxiv.org/abs/quant-ph/0110141
14. Goertzel, B.: Toward a Formal Characterization of Real-World General Intelligence. In: The Third Conference on Artificial General Intelligence (AGI-10). pp. 19-24. Atlantis Press, Amsterdam (2010) http://agi-conf.org/2010/wp-content/uploads/2009/06/paper_14.pdf
15. Tyler, T.: The Matching Pennies Project. http://matchingpennies.com/
16. Levin, L. A.: Universal Sequential Search Problems. Problems of Information Transmission **9**(3), 265-266 (1973)