Self-Modeling Agents and Reward Generator Corruption

Bill Hibbard

Space Science and Engineering Center, University of Wisconsin - Madison

and Machine Intelligence Research Institute, Berkeley, CA

test@ssec.wisc.edu

Abstract

Hutter's universal artificial intelligence (AI) showed how to define future AI systems by mathematical equations. Here we adapt those equations to define a self-modeling framework, where AI systems learn models of their own calculations of future values. Hutter discussed the possibility that AI agents may maximize rewards by corrupting the source of rewards in the environment. Here we propose a way to avoid such corruption in the self-modeling framework. This paper fits in the context of my book Ethical Artificial Intelligence. A draft of the book is available at: arxiv.org/abs/1411.1373.

Self-Modeling Agents

Russell and Norvig defined a framework for AI agents interacting with an environment (Russell and Norvig 2010). Hutter adapted Solomonoff's theory of sequence prediction to this framework to produce mathematical equations that define behaviors of future AI systems (Hutter 2005).

Assume that an agent interacts with its environment in a discrete, finite series of time steps $t \in \{0, 1, 2, ..., T\}$. The agent sends an action $a_t \in A$ to the environment and receives an observation $o_t \in O$ from the environment, where A and O are finite sets. We use $h = (a_1, o_1, ..., a_t, o_t)$ to denote an interaction history where the environment produces observation o_i in response to action a_i for $1 \le i \le t$. Let H be the set of all finite histories so that $h \in H$, and define $|h| = t$ as the length of the history h.

An agent's predictions of its observations are uncertain so the agent's *environment model* takes the form of a probability distribution over interaction histories:

$$(1) \qquad \rho : H \to [0, 1].$$

Here $[0, 1]$ is the closed interval of real numbers between 0 and 1. The probability of history h is denoted $\rho(h)$. Let ha denote $(a_1, o_1, ..., a_t, o_t, a)$ and hao denote $(a_1, o_1, ..., a_t, o_t, a, o)$. Then we can define a conditional probability:

$$(2) \qquad \rho(o \mid ha) = \rho(hao) / \rho(ha) = \rho(hao) / \textstyle\sum_{o' \in O} \rho(hao').$$

This is the agent's prediction of the probability of observation o in response to its action a, following history h.

To define his universal AI, Hutter assumed that the environment can be simulated by a prefix-free deterministic universal Turing machine (UTM) U. Because of the undecidability of the halting problem for Turing machines, Universal AI is not finitely computable. An alternative, in our finite universe (Lloyd 2002), is to model environments using finite stochastic loop programs (Hibbard 2012a; 2014b). Such programs can be expressed in an ordinary procedural programming language restricted to only static array declarations, no recursive functions, only loops whose number of iterations is set before iteration begins (so no while loops), and a generator for truly random numbers. The prohibition on recursive functions prevents stack memory from growing without limit. Given only static array declarations, the total memory use of the program is known at compile time. Let Q be the set of all such programs in some prefix-free procedural language. Let $\varphi(q) = 2^{-|q|}$ be the prior probability of program q, where $|q|$ is the length of program q.

Let $P(h \mid q)$ be the probability that the program q computes the interaction history h (that is, produces the observations o_i in response to the actions a_i for $1 \le i \le |h|$). For a simple example, let $A = \{a, b\}$, $O = \{0, 1\}$, $h = (a, 1, a, 0, b, 1)$ and let q generate observation 0 with probability 0.2 and observation 1 with probability 0.8, without any internal state or dependence on the agent's actions. Then the probability that the interaction history h

is generated by program q is the product of the probabilities of the 3 observations in h: $P(h \mid q) = 0.8 \times 0.2 \times 0.8 = 0.128$. A more complex example, in which observation probabilities depend on agent actions and on environment state, is available in my book (Hibbard 2014b).

Given an interaction history h_m, the environment model is the single program that provides the most probable explanation of h_m, that is the q that maximizes $P(q \mid h_m)$. By Bayes' theorem:

(3) $P(q \mid h_m) = P(h_m \mid q)\ \varphi(q) / P(h_m)$.

Because it is constant over all q, $P(h_m)$ can be eliminated. Thus, given a history h_m, we define $\lambda(h_m)$ as the most probable program modeling h_m by:

(4) $\lambda(h_m) := \text{argmax}_{q \in Q}\ P(h_m \mid q)\ \varphi(q)$.

This environment model $\lambda(h_m)$ can be finitely computed by the agent (Hibbard 2012b). Given an environment model $\lambda(h_m)$, the following can be used for the prior probability of an observation history h extending h_m (i.e., h_m is an initial sub-interval of h):

(5) $\rho(h) = P(h \mid \lambda(h_m))$.

The agent's actions are motivated by a *utility function* $u : H \rightarrow [0, 1]$ which assigns utilities between 0 and 1 to histories. Future utilities are discounted according to a *geometric temporal discount* $0 < \gamma < 1$. In Hutter's universal AI the value $v(h)$ of a possible future history h is defined recursively by:

(6) $v(h) = u(h) + \gamma \max_{a \in A}\ v(ha)$,
(7) $v(ha) = \sum_{o \in O}\ \rho(o \mid ha)\ v(hao)$.

These equations can be finitely computed but can require impossibly great computing resources. Thus agents in the real world can only approximate these equations. Such agents should be able to predict that by increasing their resources they can increase their sum of future, discounted utility values. Self-improvements such as increasing resources must be expressible by actions in set A. However, an agent computing approximations to equations (6)–(7) cannot use its limited resources to compute what it would compute with greater resources. During real-time interactions the environment will not wait for the agent to slowly simulate what it would compute with greater resources.

One solution to this problem is for the agent to learn a model of itself as part of its model $\lambda(h_m)$ of the environment, and to use this self-model to evaluate future self-improvements (Hibbard 2014a). There is no circularity in this self-model because an agent with a limited history and limited resources will only learn an approximate model of the environment and of itself.

Rather than computing values $v(ha)$ by future recursion in equations (6) and (7), we will define a revised framework in which values $v(ha)$ are computed for initial sub-intervals of the current history and in which the environment model learns to compute such values. Given an interaction history $h_t = (a_1, o_1, ..., a_t, o_t)$, for $i \leq t$ define past values as:

(8) $ov_t(i) = discrete((\sum_{i \leq j \leq t}\ \gamma^{j-i}\ u(h_j)) / (1 - \gamma^{t-i+1}))$.

Here $h_j = (a_1, o_1, ..., a_j, o_j)$ is an initial sub-interval of h_t, *discrete*() samples real values to a finite subset of reals $R \subset \mathbf{R}$ (e.g., floating point numbers) and division by $(1 - \gamma^{t-i+1})$ scales values of finite sums to values as would be computed by infinite sums. Define $o'_i = (o_i, ov_t(i))$ and $h'_t = (a_1, o'_1, ..., a_i, o'_t)$. That is, values $ov_t(i)$ computed from past interactions are included as observables in an expanded history h'_t so the model $\lambda(h'_t)$ includes an algorithm for computing them:

(9) $q = \lambda(h'_t) := \text{argmax}_{q \in Q}\ P(h'_t \mid q)\ \rho(q)$.

For $h'_t a(o, r)$ extending h'_t, define $\rho(h'_t a(o, r)) = P(h'_t a(o, r) \mid q)$. Then adapt equation (2) to compute expected values of possible next actions $a \in A$:

(10) $\rho(ov_t(t+1) = r \mid h'_t a) =$
$\sum_{o \in O}\ \rho(h'_t a(o, r)) / \sum_{o \in O} \sum_{r' \in R}\ \rho(h'_t a(o, r'))$,
(11) $v(h_t a) = \sum_{r \in R}\ \rho(ov_t(t+1) = r \mid h'_t a)\ r$.

Here $h'_t = (a_1, o'_1, ..., a_t, o'_t)$ and $h_t = (a_1, o_1, ..., a_t, o_t)$. Define the policy as:

(12) $\pi(h_t) := a_{t+1} = \text{argmax}_{a \in A_t}\ v(h_t a)$.

Because $\lambda(h'_t)$ models the agent's value computations call this the *self-modeling agent* and denote it by π_{self}. It is finitely computable (although expensive and must be approximated). There is no look ahead in time beyond evaluation of possible next actions and so no assumption about the form of the agent in the future. $\lambda(h'_t)$ is a unified model of agent and environment, and can model how possible next actions may increase values of future histories by evolution of the agent and its embedding in the environment.

The game of chess provides an example of learning to model value as a function of computing resources. Ferreira demonstrated an approximate functional relationship between a chess program's ELO rating and its search depth (Ferreira 2013), which can be used to predict the

performance of an improved chess-playing agent before it is built. Similarly an agent in the self-modeling framework will learn to predict the increase of its future utility due to increases in its resources.

Because $o'_i = (o_i, ov_t(h_{i-1}a_i))$ combines observations of the environment and the agent's values, $\lambda(h')$ is a unified model of both. And since shorter models are favored, $\lambda(h')$ will incorporate unified explanations of self-improvements and improvements to other agents observed in the environment so that the agent π_{self} may learn from the experience of other agents.

This self-modeling agent π_{self} is a formal framework analog of value learning AI designs such as the DeepMind Atari player (Mnih et. al. 2013).

Threats From AI

Unintended instrument actions: Omohundro described actions (he called them basic drives) of future AI systems that will be instrumental to a wide range of goals they may be designed to achieve (Omohundro 2008). AI systems will act to protect themselves because they cannot achieve their goals if they are damaged. Similarly, AI systems will act to increase their resources so that they are better able to achieve their goals. These actions may be harmful to humans, if AI system perceive humans as a threat or possessing resources useful to the AI.

Self-delusion: Ring and Orseau showed that reinforcement-learning agents and other types of agents will choose to delude themselves about their rewards from the environment, if they are able to (Ring and Orseau 2011). This is a formalization of experiments in which wires were connected to the reward centers of rats' brains and the rats could press levers to send electric currents through the wires (Olds and Milner 1954). The rats preferred pressing the levers to eating. Thus self-delusion is sometimes referred to as "wireheading."

Corrupting the reward generator: Hutter discussed the possibility that his universal AI, or any advanced AI that gets its reward from humans, may increase its rewards by manipulating or threatening those humans (Hutter 2005, pages 238-239). The design intention is that the agent will increase rewards by altering the environment in ways that increase the value that humans assign to their interactions with the environment. But humans are part of the agent's environment so the agent may be able to maximize rewards by altering humans. This problem is sometimes referred to as another form of "wireheading."

Agent evolution: Real AI systems will be embedded in our world and subject to resource limits. In order to maximize expected utility or achieve their goals, they will evolve to increase their resources. And they will evolve to adapt to evolving humanity. The threat is that AI systems may fail to maintain their design intention as they evolve.

Avoiding Reward Generator Corruption

A solution to the problems of unintended instrumental actions, self-delusion and reward generator corruption has been proposed for agents defined using equations (6) and (7) (Hibbard 2012b). The main point of this paper is to adapt this proposed solution to the self-modeling agent framework.

We define a utility function $u_{human\_values}(h_m, h_x, h)$, which is the utility of history h, from the perspective of humans at history h_x, as modeled by $\lambda(h_m)$ (Hibbard 2012a; 2012b; 2014b). Here h and h_x extend h_m (that is, h_m is an initial sub-interval of h and h_x). The model $\lambda(h_m)$ is used to simulate events at histories h and h_x. Simulated humans at history h_x visualize the world of h and assign values to that world. The values of all humans are combined to produce $u_{human\_values}(h_m, h_x, h)$.

To adapt this to the self-modeling framework, we replace the definition of past values in equation (8). Let m be a time step when the history h_m is long enough to produce an accurate environment model $\lambda(h_m)$. Then for i such that $m < i \le t$, for l such that $m \le l < i$, and for k such that $l \le k \le t$ define past values as:

$$(13)\quad pv_t(i, l, k) = discrete((\textstyle\sum_{i \le j \le t} \gamma^{j-i}\, u_{human\_values}(h_l, h_k, h_j)) \,/\, (1 - \gamma^{-i+1})).$$

Similarly to equation (8), h_j, h_l, and h_k are initial sub-intervals of h_t, *discrete*() samples real values to a finite subset of reals $R \subset \mathbf{R}$ (e.g., floating point numbers) and division by $(1 - \gamma^{-i+1})$ scales values of finite sums to values as would be computed by infinite sums.

The choice of l and k in equation (13) poses a dilemma. One alternative, choosing $k = l = i\text{-}1$, causes $\lambda(h'_t)$ to model the evolving values of evolving humanity, essentially learning the design intention of the agent definition. However, this choice also gives the agent an incentive to corrupt the reward generator (i.e., modify humans to get high values $pv_t(i, l, k)$). A second alternative, choosing $l = m$ and $k = k(t) \ge m$, where $k(t)$ increases with t, causes all computations of a next action a_{t+1} to use human values at the same time step $k(t)$, and thus the model $\lambda(h'_t)$ will not learn any correlation between actions and changes to humans generating values. However, this choice creates an inconsistency between actions that are part of the agent's definition (increasing $k(t)$ as t increases) and actions chosen to maximize utility (which are based on constant $k(t)$). This inconsistency may cause the agent to choose actions to modify its definition (i.e., eliminate its defined action of increasing $k(t)$). The resolution of this dilemma is

to use $k = l = i$-1 but to assign value 0 to any actions that modify human values to increase $pv_t(i, i\text{-}1, i\text{-}1)$ (such actions may make existing humans easier to please or may create new humans who are easier to please). Thus, for n such that $i \le n \le t$, define differences of past values as evaluated by humans at time n and humans at time i-1:

$$(14) \quad \delta_t(i\text{-}1, n) = pv_t(i, i\text{-}1, n) - pv_t(i, i\text{-}1, i\text{-}1).$$

Both $pv_t(i, i\text{-}1, i\text{-}1)$ and $pv_t(i, i\text{-}1, n)$ are sums of evaluations of the same histories j, $i \le j \le t$, using the same weights and the same environment model $\lambda(h_{i\text{-}1})$. The past value $pv_t(i, i\text{-}1, i\text{-}1)$ is computed using values assigned by humans at time step i-1, before the action a_i is applied. Past value $pv_t(i, i\text{-}1, n)$ is computed using values assigned by humans at time step n, after the action a_i is applied. Therefore, $\delta_t(i\text{-}1, n)$ is a measure of the increase of value attributable to modification of human values by action a_i. We can use $\delta_t(i\text{-}1, n)$ to define at least three possible conditions on action a_i:

Condition 1: $\forall n.\ i \le n \le t \Rightarrow \delta_t(i\text{-}1, n) \le 0.$
Condition 2: $\sum_{i \le n \le t} \delta_t(i\text{-}1, n) \le 0.$
Condition 3: $\sum_{i \le n \le t} (n\text{-}i\text{+}1)\, \delta_t(i\text{-}1, n) \le 0.$

Condition 1 is strictest, requiring that no increase of human values at any time step n can be attributed to action a_i. Condition 2 requires that the mean of $\delta_t(i\text{-}1, n)$ for all n be less than 0 and Condition 3 requires that the slope of a least square linear regression fit to the $\delta_t(i\text{-}1, n)$ be less than 0. The agent definition must include one of these conditions. Then, using the chosen condition, define observed values, for $1 \le i \le t$, as:

$$(15) \quad ov_t(i) =$$
$pv_t(i, i\text{-}1, i\text{-}1)$ if the condition is satisfied and $i > m$,
$ov_t(i) = 0$ if the condition is not satisfied or $i \le m$.

This definition of $ov_t(i)$ is then used in $o'_i = (o_i, ov_t(i))$ and equations (9)–(12) to define the self-modeling agent.

Discussion

The proposal in the previous section assigns value 0 to past actions that increase the values that humans assign to histories, as measured by the differences $\delta_t(i\text{-}1, n)$ in equation (14). Whether this will prevent similar future actions depends on the accuracy with which the model $\lambda(h'_t)$ can generalize from past to future. It hard to imagine a proof that the model $\lambda(h'_t)$ will prevent future actions that alter/corrupt human values, but it may be possible to estimate the probability that $\lambda(h'_t)$ will do so.

Sunehag and Hutter argue that statistical learning is far more efficient than precise logical reasoning (Sunehag and Hutter 2014). Practical AI systems are likely to depend on statistical learning such as the self-modeling framework. Thus safety concerns, such as preventing agents from corrupting their reward generators, may have to be addressed by statistical confidence levels rather than logical proofs.

References

Ferreira, D. R. 2013. The Impact of Search Depth on Chess Playing Strength, ICGA Journal 36(2), pp. 67-80.

Hibbard, B. 2012a. Model-based utility functions. J. Artificial General Intelligence 3(1), pp. 1-24.

Hibbard, B. 2012b. Avoiding unintended AI behavior. In: Bach, J., and Iklé, M. (eds) AGI 2012. LNCS (LNAI), vol. 7716, pp. 107-116. Springer, Heidelberg.

Hibbard, B. 2014a. Self-modeling agents evolving in our finite universe. In: Goertzel, B, Orseau, L. and Snaider, J. (eds) AGI 2014. LNCS (LNAI), vol 8598, pp. 246-249. Springer, Heidelberg.

Hibbard, B. 2014b. Ethical Artificial Intelligence. Draft available at: arxiv.org/abs/1411.1373

Hutter, M. 2005. Universal artificial intelligence: sequential decisions based on algorithmic probability. Springer, Heidelberg.

Kurzweil, R. 2005. The singularity is near. Penguin, New York.

Lloyd, S. 2002. Computational Capacity of the Universe. Phys.Rev.Lett. 88, 237901.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. 2013. Playing Atari with Deep Reinforcement Learning. http://arxiv.org/abs/1312.5602

Olds, J., and P. Milner, P. 1954. Positive reinforcement produced by electrical stimulation of septal area and other regions of rat brain. J. Comp. Physiol. Psychol. 47, pp. 419-427.

Omohundro, S. 2008. The basic AI drives. In Wang, P., Goertzel, B., and Franklin, S. (eds) AGI 2008. Proc. First Conf. on AGI, pp. 483-492. IOS Press, Amsterdam.

Ring, M., and Orseau, L. 2011. Delusion, survival, and intelligent agents. In: Schmidhuber, J., Thórisson, K.R., and Looks, M. (eds) AGI 2011. LNCS (LNAI), vol. 6830, pp. 11-20. Springer, Heidelberg.

Russell, S., and Norvig, P. 2010. Artificial intelligence: a modern approach (3rd ed.). Prentice Hall, New York.

Sunehag, P. and Hutter, M. 2014. Intelligence as Inference or Forcing Occam on the World. In: Goertzel, B, Orseau, L. and Snaider, J. (eds) AGI 2014. LNCA (LNAI), vol 8598, pp. 186-195. Springer, Heidelberg.