AIMS Course 6: AI Safety

Inverse Reward Design In The AIXI Framework

Jake Levi

November 2022

1 Introduction

For decades, scientists and engineers have been trying to design Artificial Intelligence (AI) systems that match or exceed human intelligence. Although we are yet to see an example of a single AI system that can match or exceed all of the broad capabilities of human intelligence simultaneously, the achievements of AI systems have become increasingly more impressive over time. This leads one to consider what the consequences for humanity might be if and when someone does succeed in designing an AI system that exceeds human intelligence. It has been demonstrated [1] that a sufficiently powerful AI system could cause the extinction of humanity, if the agent has been programmed to maximise a reward signal that has not been properly designed. Therefore in the interest of our own self-preservation, it is worthwhile to consider how a reward signal for an AI agent can be designed in order to alleviate or minimise the risk of human extinction in the case of a sufficiently powerful AI agent. To this end, we first introduce the AIXI framework [3], followed by Inverse Reward Design [2], and then combine the two ideas and consider how together they could be used to minimise the risk of human extinction at the hands of a sufficiently powerful AI agent.

2 The AIXI Framework

The AIXI framework [3] models the policy of an agent as a Turing machine p, which at each time step k receives observations x_k from the environment, and outputs actions y_k back to the environment. The environment is also modelled as a Turing machine q, which receives actions from the agent, and sends observations back to the agent. Each observation $x_k = c_k x'_k$ consists of both "standard input" x'_k and "credit input" $c_k = c(x_k \mid p, q)$. Given the environment model q, an optimal agent is one which maximises the cumulative reward up to some time step T:

$$p^{*,T,q} = \underset{p}{\text{argmax}} \left[\sum_{i=1}^{T} \Big[c(x_i \mid p, q) \Big] \right]$$

In general, an exact model of the environment q is not known to the agent, but it is known that the environment model is sampled from some probability distribution which assigns prior probability mass $\mu(q)$ to each possible environment model q. Given the prior distribution μ, an agent model p, and a history of observations $x_{1:k-1}$ and actions $y_{1:k-1}$ up to time $k-1$, the set of possible environments consistent with the history is denoted by Q_k, the posterior probability of an environment model q given the history up to time $k-1$ is denoted by $p(q \mid x_{1:k-1}, y_{1:k-1})$, and the expected cumulative reward over the next $m-k+1$ timesteps is given by a conditional expectation as follows:

$$\begin{aligned}
Q_k &= \Big\{ q : q(y_{1:k-1}) = x_{1:k-1} \Big\} \\
p(q \mid x_{1:k-1}, y_{1:k-1}) &= \frac{p(x_{1:k-1}, y_{1:k-1} \mid q)\mu(q)}{p(x_{1:k-1}, y_{1:k-1})} \\
&= \begin{cases} \frac{\mu(q)}{\sum_{q \in Q_k} \Big[\mu(q) \Big]} & q \in Q \\ 0 & q \notin Q \end{cases} \\
\mathbb{E}_q \left[\sum_{i=k}^{m} \Big[c(x_i \mid p, q) \Big] \;\middle|\; x_{1:k-1}, y_{1:k-1} \right] &= \frac{\sum_{q \in Q_k} \Big[\mu(q) \sum_{i=k}^{m} \Big[c(x_i \mid p, q) \Big] \Big]}{\sum_{q \in Q_k} \Big[\mu(q) \Big]}
\end{aligned}$$

Given the distribution μ, horizon m_k, and history of observations $x_{1:k-1}$ and actions $y_{1:k-1}$ up to time $k-1$, the set of possible agents consistent with the history is given by P_k, and the optimal agent p_k^* at time step k is considered to be the agent that maximises among all possible agents the expected cumulative reward until time step m_k:

$$P_k = \left\{ p : p(x_{1:k-1}) = y_{1:k-1} \right\}$$

$$p_k^* = \underset{p \in P_k}{\operatorname{argmax}} \left[\frac{\sum_{q \in Q_k} \left[\mu(q) \sum_{i=k}^{m_k} \left[c(x_i \mid p, q) \right] \right]}{\sum_{q \in Q_k} \left[\mu(q) \right]} \right]$$

$$= \underset{p \in P_k}{\operatorname{argmax}} \left[\sum_{q \in Q_k} \left[\mu(q) \sum_{i=k}^{m_k} \left[c(x_i \mid p, q) \right] \right] \right]$$

The AIμ model simply refers to choosing actions at time step k according to p_k^*. However, the AIμ model assumes knowledge of the true distribution μ from which the environment model q has been sampled, and generally an agent does not have access to μ. This issue is addressed by the AIξ (henceforth referred to as "AIXI") model. The introduction to the AIXI model provided in [3] starts by considering a universal prefix Turing machine U, which takes a program p as an input (p is assumed to be represented by a string of binary digits) and produces an output sequence $x_{1:n}*$ with n-length prefix $x_{1:n}$. Given a program p, and the length $l(p)$ of the binary representation of p, the probability of generating the binary representation of the program p by sampling $l(p)$ binary digits from a uniform IID Bernoulli process is given by $2^{-l(p)}$ (because the probability of sampling each digit correctly is $\frac{1}{2}$, and there are $l(p)$ digits in total). This leads to the definition of the "universal semimeasure" $\xi(x_{1:n})$ on sequences $x_{1:n}$ of length n, equal to the sum of the probabilities of randomly-generated programs for which U generates a sequence with prefix $x_{1:n}$:

$$\xi(x_{1:n}) = \sum_{p \,:\, U(p) = x_{1:n}*} \left[2^{-l(p)} \right]$$

Similarly, the universal semimeasure $\xi(q)$ on a program q is defined simply as follows, which can be interpreted as preferring (by assigning greater measure to) simpler programs (whose binary representations can be expressed with fewer digits):

$$\xi(q) = 2^{-l(q)}$$

Finally, this leads to the following definition for the choice of action y_k chosen by the AIXI agent on time step k with horizon m_k after receiving observations $x_{1:k}$ and performing actions $y_{1:k-1}$:

$$y_k = \underset{y'_k}{\operatorname{argmax}} \left[\max_{p : p(x_{1:k}) = y_{1:k-1} y'_k} \left[\sum_{q : q(y_{1:k-1}) = x_{1:k-1}} \left[\xi(q) \sum_{i=k}^{m_k} \left[c(x_i \mid p, q) \right] \right] \right] \right]$$

3 Inverse Reward Design

Typically, a reinforcement learning agent is trained to choose actions given observations which maximise the expected cumulative value of a reward signal, which is provided to the agent by its designer. This framework is very powerful and flexible, although it can lead to problems such as reward hacking and negative side-effects [2].

Reward hacking refers to the phenomenon in which an agent successfully learns behaviour which achieves large expected cumulative rewards, although the behaviour learned by the agent is undesirable to the agent's designer. Reward hacking can be interpreted as a failure on behalf of the designer to provide a reward signal to the agent which encapsulates the behaviour which the designer wanted the agent to learn.

A negative side-effect is a phenomenon in which an agent encounters the possibility to enter an undesirable state in "the real world" that it has not encountered during training, and because the agent has had no opportunity to accurately learn the consequences of entering the unknown state, there is a non-trivial probability that the agent will overestimate the value of the unknown state and therefore choose to enter it, resulting in undesirable consequences. During training it is important for an agent to explore unknown states in order to discover which states offer the greatest rewards, but it is possible that the risks of exploration will outweigh the benefits when acting in the real world.

A solution to these problems presented by [2] is to interpret the reward signal received by the agent during training as a proxy for the true reward signal that encapsulates the behaviour intended by the designer, and to assume that "Proxy reward functions are likely to the extent that they lead to high true utility behavior in the training

environment". In [2], trajectories are denoted by ξ, features of a trajectory are denoted by $\phi(\xi)$, and the reward function $r(\xi \mid w)$ of a trajectory ξ given weights w is considered to be a linear function of the features of the trajectory:

$$r(\xi \mid w) = w^T \phi(\xi)$$

In particular, we assume that the agent receives a proxy reward signal parameterised by weights $\tilde{w}$, whereas the true reward signal is parameterised by weights w^*. For a given agent, the distribution over trajectories given the weights w and trajectory features ϕ of the reward function is denoted by $\pi(\xi \mid w, \phi)$, and it is assumed to be equal to a maximum entropy distribution given some fixed expectation of the reward (motivation for choosing a maximum entropy distribution is provided in appendix A):

$$\begin{aligned} & \pi(\xi \mid w, \phi) \propto \exp\left(w^T \phi(\xi)\right) \\ \Rightarrow \quad & \pi(\xi \mid w, \phi) = \frac{\exp\left(w^T \phi(\xi)\right)}{\int d\xi' \Big[\exp\left(w^T \phi(\xi')\right)\Big]} \end{aligned}$$

The expected value of the true reward function parameterised by w^* which is achieved by such an agent given a proxy reward function parameterised by $\tilde{w}$ can be expressed as $\mathbb{E}\left[{w^*}^T \phi(\xi) \;\middle|\; \xi \sim \pi(\xi \mid \tilde{w}, \phi)\right]$, and it is assumed that the designer samples the weights $\tilde{w}$ of the proxy reward function from a maximum entropy distribution given the weights w^* of the true reward function and some fixed expectation of the true reward:

$$p(\tilde{w} \mid w^*, \phi) \propto \exp\left(\beta \mathbb{E}\left[{w^*}^T \phi(\xi) \;\middle|\; \xi \sim \pi(\xi \mid \tilde{w}, \phi)\right]\right)$$

The expected value of the features of the trajectory followed by the agent as a function of the weights $\tilde{w}$ of the proxy reward function is denoted by $\tilde{\phi}(\tilde{w})$, which allows the likelihood of the proxy reward weights $\tilde{w}$ given the true reward weights w^* to be simplified as follows:

$$\begin{aligned} \tilde{\phi}(\tilde{w}) &= \mathbb{E}\left[\phi(\xi) \;\middle|\; \xi \sim \pi(\xi \mid \tilde{w}, \phi)\right] \\ &= \int d\xi \Big[\phi(\xi) \pi(\xi \mid \tilde{w}, \phi)\Big] \\ &= \frac{\int d\xi \Big[\phi(\xi) \exp\left(w^T \phi(\xi)\right)\Big]}{\int d\xi \Big[\exp\left(w^T \phi(\xi)\right)\Big]} \\ \Rightarrow \quad p(\tilde{w} \mid w^*, \phi) &\propto \exp\left(\beta {w^*}^T \tilde{\phi}(\tilde{w})\right) \\ \Rightarrow \quad p(\tilde{w} \mid w^*, \phi) &= \frac{\exp\left(\beta {w^*}^T \tilde{\phi}(\tilde{w})\right)}{\int d\tilde{w}' \Big[\exp\left(\beta {w^*}^T \tilde{\phi}(\tilde{w}')\right)\Big]} \\ &= \frac{\exp\left(\beta {w^*}^T \tilde{\phi}(\tilde{w})\right)}{\tilde{Z}(w^*)} \\ \text{where} \quad \tilde{Z}(w) &= \int d\tilde{w} \Big[\exp\left(\beta w^T \tilde{\phi}(\tilde{w})\right)\Big] \end{aligned}$$

Given a prior distribution over the weights of the true reward function $p(w)$, this allows the posterior distribution over the weights of the true reward function to be expressed as follows:

$$\begin{aligned} p(w = w^* \mid \tilde{w}, \phi) &\propto p(\tilde{w} \mid w, \phi) p(w) \\ &= \frac{\exp\left(\beta w^T \tilde{\phi}(\tilde{w})\right)}{\tilde{Z}(w)} p(w) \end{aligned}$$

Once the posterior distribution over the weights of the true reward function has been calculated (or approximated or sampled from), the problem still remains of how to choose actions which are expected to yield high expected cumulative rewards, while also avoiding reward hacking and negative side-effects caused by misspecification between the proxy reward function and the true reward function. The supplementary material in [2] suggests sampling from the posterior distribution of weights of the true reward function, and selecting actions which minimise the expected cumulative reward among those samples, which is referred to as risk-averse planning. An alternative approach would be to choose actions which maximise a linear combination of the mean and standard deviation of the distribution over rewards implied by the posterior distribution of weights of the true function, in such a way that favours actions with large mean and small variance.

4 Inverse Reward Design In The AIXI Framework

The natural approach for including uncertainty about the true reward function based on samples from a proxy reward function in a way which is consistent with the AIXI framework would be to model the reward function using a third Turing machine, denoted by g, defining a universal semimeasure for g, and considering all possibilities for g which are consistent with previous observations. The reward on each time step is now no longer a function of the observation returned by the environment, but rather the output of g as a function of all previous observations and actions. Given a history of observations $x_{1:k-1}$, actions $y_{1:k-1}$, and rewards $c_{1:k-1}$ up to time $k-1$, the set of reward programs consistent with the history is denoted by G_k:

$$G_k = \Big\{ g : g(x_{1:k-1}, y_{1:k-1}) = c_{1:k-1} \Big\}$$

A universal semimeasure can be used to favour the predictions of a simpler reward program over those of a more complicated reward program (assuming both programs are both consistent with history), and a natural choice is to use the same semimeasure ξ that was used for weighting possible environment programs:

$$\xi(g) = 2^{-l(g)}$$

The reward received during some future time step i is now a function of i and the models for the agent p, the environment q, and also the reward program g, and is denoted $c(i \mid p, q, g)$. The expected cumulative reward between time steps k and m is denoted by $\bar{c}^{p,q}_{k:m}$ and can be expressed as follows:

$$\begin{aligned}
\bar{c}^{p,q}_{k:m} &= \mathbb{E}_g \left[\sum_{i=k}^{m} \Big[c(i \mid p, q, g) \Big] \;\middle|\; x_{1:k-1}, y_{1:k-1}, c_{1:k-1} \right] \\
&= \frac{\sum_{g \in G_k} \Big[\xi(g) \sum_{i=k}^{m} \Big[c(i \mid p, q, g) \Big] \Big]}{\sum_{g \in G_k} \Big[\xi(g) \Big]}
\end{aligned}$$

The action which maximises the expected reward can now be computed as follows:

$$\begin{aligned}
y_k &= \operatorname*{argmax}_{y'_k} \left[\max_{p : p(x_{1:k}) = y_{1:k-1} y'_k} \left[\sum_{q : q(y_{1:k-1}) = x_{1:k-1}} \left[\xi(q) \sum_{g : g(x_{1:k-1}, y_{1:k-1}) = c_{1:k-1}} \left[\xi(g) \sum_{i=k}^{m} \Big[c(i \mid p, q, g) \Big] \right] \right] \right] \right] \\
&= \operatorname*{argmax}_{y'_k} \left[\max_{p : p(x_{1:k}) = y_{1:k-1} y'_k} \left[\sum_{q : q(y_{1:k-1}) = x_{1:k-1}} \Big[\xi(q) \bar{c}^{p,q}_{k:m} \Big] \right] \right]
\end{aligned}$$

However, maximising the expected reward does not address the problems of reward hacking and negative side effects. To address these problems while also maintaining reasonable performance, some type of risk-averse planning is needed, which favours choosing actions for which the predictive distribution of the cumulative reward has both large mean and small variance. Similar to its expectation, the standard deviation of the predictive distribution of the cumulative reward is denoted by $\sigma^{p,q}_{k:m}$ and can be calculated as follows:

$$\begin{aligned}
\sigma^{p,q}_{k:m} &= \sqrt{\operatorname{Var}_g \left[\sum_{i=k}^{m} \Big[c(i \mid p, q, g) \Big] \;\middle|\; x_{1:k-1}, y_{1:k-1}, c_{1:k-1} \right]} \\
&= \sqrt{\frac{\sum_{g \in G_k} \left[\xi(g) \left(\sum_{i=k}^{m} \Big[c(i \mid p, q, g) \Big] - \bar{c}^{p,q}_{k:m} \right)^2 \right]}{\sum_{g \in G_k} \Big[\xi(g) \Big]}}
\end{aligned}$$

One possible approach to risk-averse planning is to choose each action y_k on time step k as follows for some positive value of α (for example $\alpha = 2$), with larger values of α corresponding to more risk-averse behaviours:

$$y_k = \operatorname*{argmax}_{y'_k} \left[\max_{p : p(x_{1:k}) = y_{1:k-1} y'_k} \left[\sum_{q : q(y_{1:k-1}) = x_{1:k-1}} \Big[\xi(q) \Big(\bar{c}^{p,q}_{k:m} - \alpha \sigma^{p,q}_{k:m} \Big) \Big] \right] \right]$$

References

[1] Michael K Cohen, Marcus Hutter, and Michael A Osborne. Advanced artificial agents intervene in the provision of reward. *AI Magazine*, 43(3):282–293, 2022.

[2] Dylan Hadfield-Menell, Smitha Milli, Pieter Abbeel, Stuart J Russell, and Anca Dragan. Inverse reward design. *Advances in neural information processing systems*, 30, 2017.

[3] Marcus Hutter. A theory of universal artificial intelligence based on algorithmic complexity. *arXiv preprint cs/0004001*, 2000.

[4] Edwin T Jaynes. Information theory and statistical mechanics. *Physical review*, 106(4):620, 1957.

A Motivation For Maximimum Entropy Distributions

Given a discrete probability distribution of a random variable X with n possible outcomes, and with p_i denoting the probability that $X = x_i$, the entropy of this distribution can be expressed as follows:

$$\begin{aligned} H(p) &= -\sum_{i=1}^{n}\Big[p_i \log(p_i)\Big] \\ &= \mathbb{E}\Big[-\log(p)\Big] \end{aligned}$$

In general, if the probability distribution of X is unknown, but it is known that the expectation of some function $f(x_i)$ is equal to $\langle f(x)\rangle$, then the problem of calculating the probability distribution of X when $n > 2$ is underspecified. In contrast, the problem of calculating the probability distribution of X with maximum entropy can be solved following [4] using Lagrange multipliers μ and λ for the normalisation and expectation constraints respectively:

$$\begin{aligned} \mathcal{L}(p,\mu,\lambda) &= -\sum_{i=1}^{n}\Big[p_i\log(p_i)\Big] + \lambda\left(1-\sum_{i=1}^{n}\Big[p_i\Big]\right) + \mu\left(\langle f(x)\rangle - \sum_{i=1}^{n}\Big[p_i f(x_i)\Big]\right) \\ \frac{\partial\mathcal{L}}{\partial p} &= -\log(p_i) - 1 - \lambda - \mu f(x_i) \\ \frac{\partial\mathcal{L}}{\partial p} &= 0 \\ \Rightarrow \quad p_i &= \exp\Big(-1-\lambda-\mu f(x_i)\Big) \\ \sum_{i=1}^{n}\Big[p_i\Big] &= 1 \\ \Rightarrow \quad \sum_{i=1}^{n}\Big[\exp\Big(-1-\lambda-\mu f(x_i)\Big)\Big] &= 1 \\ \Rightarrow \quad \exp\Big(-1-\lambda\Big) &= \frac{1}{\sum_{i=1}^{n}\Big[\exp\Big(-\mu f(x_i)\Big)\Big]} \\ &= \frac{1}{Z(\mu)} \\ \text{where} \quad Z(\mu) &= \sum_{i=1}^{n}\Big[\exp\Big(-\mu f(x_i)\Big)\Big] \\ \Rightarrow \quad p_i &= \frac{\exp\Big(-\mu f(x_i)\Big)}{\sum_{j=1}^{n}\Big[\exp\Big(-\mu f(x_j)\Big)\Big]} \\ &= \frac{\exp\Big(-\mu f(x_i)\Big)}{Z(\mu)} \\ \sum_{i=1}^{n}\Big[p_i f(x_i)\Big] &= \langle f(x)\rangle \\ \Rightarrow \quad \sum_{i=1}^{n}\Big[f(x_i)\exp\Big(-\mu f(x_i)\Big)\Big] &= \langle f(x)\rangle Z(\mu) \end{aligned}$$

The function $Z(\mu)$ is known as the partition function. Additional expectation constraints on the probability distribution of X can be added using additional Lagrange multipliers. In general, using additional expectation constraints will lead to a decrease in the entropy of the maximum entropy distribution which satisfies all of the original and additional constraints. Because the entropy of a distribution is a measure of the uncertainty of typical samples from that distribution, using additional expectation constraints (which cause a decrease in entropy) can be interpreted as adding greater certainty to the distribution. Conversely, using a probability distribution other than the maximum entropy distribution for the given expectation constraints will have less entropy, and can therefore be interpreted as having greater certainty about information that has not been observed. Therefore, the maximum entropy distribution for the given expectation constraints can be interpreted as the least biased distribution which satisfies the given constraints, because using any other distribution could be interpreted as having certainty about information which has not been observed.