The Workshops of the Thirtieth AAAI Conference on Artificial Intelligence
AI, Ethics, and Society: Technical Report WS-16-02

Quantilizers: A Safer Alternative to Maximizers for Limited Optimization

Jessica Taylor
Machine Intelligence Research Institute
jessica@intelligence.org

Abstract

In the field of AI, *expected utility maximizers* are commonly used as a model for idealized agents. However, expected utility maximization can lead to unintended solutions when the utility function does not quantify everything the operators care about: imagine, for example, an expected utility maximizer tasked with winning money on the stock market, which has no regard for whether it accidentally causes a market crash. Once AI systems become sufficiently intelligent and powerful, these unintended solutions could become quite dangerous. In this paper, we describe an alternative to expected utility maximization for powerful AI systems, which we call *expected utility quantilization*. This could allow the construction of AI systems that do not necessarily fall into strange and unanticipated shortcuts and edge cases in pursuit of their goals.

1 Expected Utility Maximization

Many frameworks for studying idealized agents assume that agents attempt to maximize the expectation of some utility function, as seen in the works of Heckerman and Horvitz (1990), Bacchus and Grove (1995), and Chajewska, Koller, and Parr (2000). von Neumann and Morgenstern (1944) provide compelling arguments that any rational agent with consistent preferences must act as if it is maximizing some utility function, and Russell and Norvig (2010, chap. 16) use the expected utility maximization framework extensively as the basic model of an idealized rational agent. Formally, let $\mathcal{A}$ be the finite[1] set of available actions, $\mathcal{O}$ the set of possible outcomes, and $W : \mathcal{A} \to \Delta\mathcal{O}$ map each action to the predicted outcome distribution resulting from that action[2]. Let $U : \mathcal{O} \to [0, 1]$ be a utility function, with $U(o)$ being the utility of outcome o. Then an expected utility maximizer is an agent that chooses an action $a \in \mathcal{A}$ that maximizes $\mathbb{E}\left[U\left(W\left(a\right)\right)\right]$.

We make no argument against expected utility maximization on the grounds of rationality. However, maximizing the expectation of some utility function could produce large unintended consequences whenever U does not accurately capture all the relevant criteria. Some unintended consequences of this form can already be observed in modern AI systems. For example, consider the genetic algorithm used by Nguyen, Yosinski, and Clune (2014) to generate an image which would be classified by a deep neural network as a starfish, with extremely high confidence. The resulting image ended up completely unrecognizable, looking nothing at all like a starfish.

Of course, Nguyen, Yosinski, and Clune (2014) intended to develop images that would be misclassified, but this demonstrates the point that an algorithm directed to find the image that *most* classified as a starfish, generated an image that was very strange indeed.

Another example of unintended solutions resulting from maximization is given by Thompson (1997), who used genetic algorithms to "artificially evolve" a circuit that could differentiate between two different inputs via a 10 x 10 field programmable gate array (FPGA). The resulting circuit worked, using only 21 of the 100 available FPGA cells. Curiously, five of those cells were not connected to either the input or the output—the evolved circuit was making use of them via the physical features of that specific chip and machine (Thompson suggest that the circuit was taking advantage of electromagnetic coupling or power-supply loading). If the goal was to design a circuit that could also work on other chips, then this "maximization" process (of finding a very small algorithm that, on that one circuit board, was good at distinguishing between two inputs) produced an unintended result that was dependent upon physical features of the chip that the designers thought were irrelevant. The resulting circuit performed well by the test metric, but the result would not likely have been usable on any other physical chips.

Unintended solutions of this form could become quite dangerous in highly autonomous artificially intelligent systems with capabilities that strongly exceed those of humans. Bostrom (2014) discusses a number of potential dangers stemming from expected utility maximization in powerful agents. He considers "superintelligent" agents, defined as agents that are "smarter than the best human brains in practically every field," and describes the problem of *perverse instantiation*, his term for the seemingly-perverse unintended consequences of directing a powerful agent to attempt to maximize a utility

[1] Expected utility maximizers and quantilizers can also be defined for infinite set of actions, but in this paper we discuss only finite action sets for simplicity.

[2] Defining W is not straightforward. See Rosenbaum and Rubin (1983) for a formalization of the "causal effects" of an action, whose maximization results in causal decision theory; see Soares and Fallenstein (2015) for discussion of alternative decision theories.

function which is not in fact aligned with our goals. For example, Bostrom (2014) illustrates this by describing a machine that, when told to make humans smile, accomplishes this goal by paralyzing human faces into permanent smiles, rather than by causing us to smile via the usual means. Another example is given by Soares (2015), who describes an agent directed to "cure cancer" which attempts to kidnap human test subjects. Clearly, before an autonomous system can be granted great autonomy and power, we must have some assurances that it will not, in pursuit of its goals, produce any such "perverse" unintended consequences.

In both the short term and the long term, the problem with expected utility maximization is that the system's utility function likely will not capture all the complexities of human value, as humans care about many features of the environment that are difficult to capture in any simple utility function (Soares 2015).

Of course, no practical system acting in the real world can actually *maximize* expected utility, as finding the literally optimal policy is wildly intractable. Nevertheless, it is common practice to design systems that *approximate* expected utility maximization, and insofar as expected utility maximization would be unsatisfactory, such systems would become more and more dangerous as algorithms and approximations improve. If we are to design AI systems which safely pursue simple goals, an alternative may be necessary.

2 Expected Utility Quantilization

Given that utility maximization can have many unintended side effects, Armstrong, Sandberg, and Bostrom (2012) and others have suggested designing systems that perform some sort of "limited optimization," that is, systems which achieve their goals in some non-extreme way, without significantly disrupting anything outside the system or otherwise disturbing the normal course of events. For example, intuition says it should be possible to direct a powerful AI system to "just make paperclips" in such a way that it runs a successful paperclip factory, without ever attempting to turn as much of the universe as possible into paperclips.

Simon (1956) propose instead designing expected utility "satisficers," agents that choose any action that achieves an expected utility above some fixed threshold. In our notation, an expected utility satisficer is one that chooses an action from the set $\{a \in \mathcal{A} \colon \mathbb{E}\left[U\left(W\left(a\right)\right)\right] \geq t\}$ for some threshold t. (Indeed, Fallenstein and Soares (2014) describe a toy model of satisficing agents in simple environments.) However, the satisficing framework is under-defined and not necessarily satisfactory. When it comes to powerful intelligent agents, it may often be that the easiest way to satisfice is to maximize: if the paperclip AI system can easily create a sub-agent that tries as hard as it can to make paperclips, it may be that the easiest satisficing action is simply to construct a sub-agent that attempts to approximate something like expected utility maximization instead.

Abbeel and Ng (2004) have suggested *apprenticeship learning* as an alternative to maximization when it is difficult to explicitly specify the correct goal. In apprenticeship learning, an AI system learns to closely imitate a human expert performing a task. An agent which only executes actions that its operators would have executed seems likely to be immune to Bostrom's problem of perverse instantiation. Yet while mimicking humans avoids many unintended effects of maximization, a system that merely mimics humans cannot easily surpass the performance of those it is mimicking (except perhaps by performing the same actions faster and more reliably). Such a technique could prove quite useful, but wouldn't help us design algorithms intended to outperform humans by finding plans, strategies, and techniques that the operators would not have been able to identify.

In this paper, we propose *quantilization*, which interpolates between mimicry and maximization in order to gain some advantages of both.

Definition 1 (Quantilizer)**.** *A q-quantilizer is an agent that, when faced with a decision problem, returns a random action in the top q proportion of some "base distribution" over actions, sorted by the expected utility achieved if that action is executed.*

Intuitively, a 0.1-quantilizer (which we could also call a "ten percentilizer") selects a random action from the top 10% of actions. More formally, let A be the action set and γ be some distribution over actions. For example, γ could be uniformly random, or it could be a distribution estimated from human actions. Let $f : [0, 1] \to \mathcal{A}$ be a function that represents γ, but sorted by expected utility:

$$\begin{gathered}\forall x, y \in [0,1] \\ x > y \implies \mathbb{E}\left[U(W(f(x)))\right] \geq \mathbb{E}\left[U(W(f(y)))\right] \\ \mu(\{x \in [0,1] \mid f(x) = a\}) = \gamma(a)\end{gathered}$$

where μ is the Lebesgue measure on the reals. For any $0 < q \leq 1$, a q-quantilizer samples a number x from the uniform distribution over $[1-q, 1]$ and returns $f(x)$. We will write $Q_{U,W,\gamma,q}(a)$ for the probability that it returns action a, where U is the quantilizer's utility function.[3] We will abbreviate this as simply $Q_q(a)$, where it is understood that Q_q is parameterized on U, W, and γ.

As q approaches 0, a q-quantilizer reduces to a maximizer whose available action set is the support of γ. A 0.01-quantilizer (also known as a "one percentilizer") samples a random action from the top 1%; a 0.5-quantilizer ("fifty-percentilizer") samples a random above-average action; a 1-quantilizer just mimics γ.

As an example, suppose there are 3 actions $\mathcal{A} = \{a, b, c\}$ with expected utilities

$$\begin{aligned}\mathbb{E}\left[U(W(a))\right] &= 0.2 \\ \mathbb{E}\left[U(W(b))\right] &= 0.5 \\ \mathbb{E}\left[U(W(c))\right] &= 0.7\end{aligned}$$

as illustrated in figure 1. A maximizer will always return action c. If we define a uniform base distribution γ, then the behavior of the q-quantilizer Q_q is:

[3] Since there are multiple sortings of the actions f, some leading to different action distributions, this notation is imprecise. To make the notation more precise, we could specify some canonical way of defining f, such as assigning as much probability mass as possible to lexicographically earlier actions.

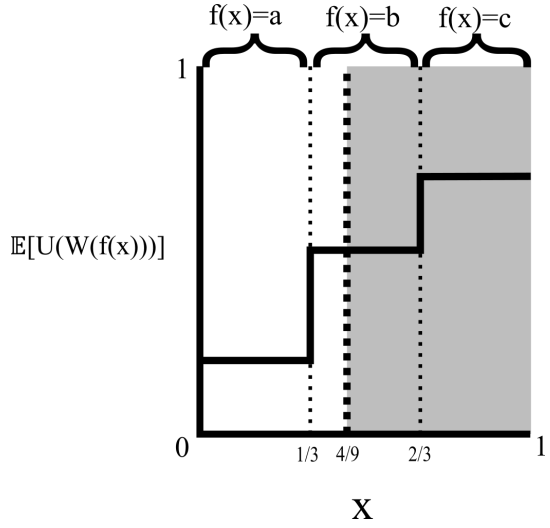

Figure 1: A representation of a $5/9$-quantilizer for the three-action example. The quantilizer samples x from the gray region and returns the corresponding action $f(x)$, so it returns b with $2/5$ probability and c with $3/5$ probability.

- If $q \leq \frac{1}{3}$, it always returns c.
- If $\frac{1}{3} \leq q \leq \frac{2}{3}$, then it returns c with probability $\frac{1}{3q}$ and b the rest of the time.
- If $q \geq \frac{2}{3}$, then it returns c with probability $\frac{1}{3q}$, b with probability $\frac{1}{3q}$, and a otherwise.

In this example, it is clear that the quantilizer interpolates between maximizing and mimicking γ depending on q.

2.1 Derivation from a Conservative Cost Constraint

We will now explore a few circumstances under which one might prefer to design a system that quantilizes expected utility, rather than maximizing it. Definition 1 gives some hint as to what sort of situation quantilizers are designed for. Given some trusted base distribution γ of actions, such as a generative model for the actions that a human might take, the distribution $Q_{0.01}$ is roughly the distribution of actions that a human might take if they were attempting to score well by U (but were otherwise adhering to "normal" behavior). That is, if γ puts most of its measure on actions that humans might generate when solving similar problems, then $Q_{0.01}$ is similar to the distribution of actions a human might take conditional on that action having a high utility according to U. Thus, intuitively, a 0.01-quantilizer is designed to solve problems in a manner similar to how a human would solve that problem.

Furthermore, under certain assumptions, we can prove that building a quantilizer instead of a maximizer is not too costly to our true utility function, in the sense that running the quantilizer won't be much worse than simply sampling from the base distribution γ.

To show this, assume that an AI system is built to quantilize expected utility for some utility function U. Assume that our true utility function is some other function V. Intuitively, we will imagine that for most actions, $\mathbb{E}\left[U(W(a))\right]$ is similar to $\mathbb{E}\left[V(W(a))\right]$, but for the actions a^U which *maximize* $\mathbb{E}\left[U(W(a^U))\right]$ (or come close to maximizing it), the value of $\mathbb{E}\left[V(W(a^U))\right]$ is much lower (because a^U is some edge case or "perverse instantiation," such as generating smiles by paralyzing people's faces). Thus, as in Swaminathan and Joachims (2015), we assume there is some unknown cost function $c\colon \mathcal{A} \to \mathbb{R}_{\geq 0}$, which describes the negative contribution of an action a to V. For example, if there is some action `noop` which the AI system may execute immediately to shut down without having any further effect on the world, then we could define $c(a) = \max\{0, \mathbb{E}\left[V(W(\texttt{noop}))\right] - \mathbb{E}\left[V(W(a))\right]\}$. We expect c to be quite difficult to approximate, because human values are complex and depend on many difficult-to-quantify aspects of the world. Indeed, if we could formally encode c, we could build it into U, and then we would have much less need to worry about perverse instantiations. Even without an approximation of c, though, we can still prove that under certain assumptions, a quantilizer will not incur a high cost according to c.

Assume that the cost of a random action from γ is no more than 1. This assumption may be justified if, for example, γ is a uniformly random action and uniformly random actions are highly unlikely to be harmful. Write $\mathbb{E}_{A\sim\gamma}\left[c(A)\right] \leq 1$ for the expected cost of a random action sampled from γ. It is now possible to bound the cost of an action sampled from Q_q:

Lemma 1 (Cost bound on quantilizers)**.**

$$\mathbb{E}_{A\sim Q_q}\left[c(A)\right] \leq \frac{1}{q}$$

Proof. Observe that Q_q "fits under" γ when scaled down by a factor of q:

$$Q_q(a) = \frac{1}{q}\mu(\{x \in [1-q, 1] \mid f(x) = a\}) \leq \frac{1}{q}\gamma(a).$$

Therefore, it is possible to bound the cost of Q_q:

$$\begin{aligned}\mathbb{E}_{A\sim Q_q}\left[c(A)\right] &= \sum_{a\in\mathcal{A}} Q_q(a)c(a) \\ &\leq \sum_{a\in\mathcal{A}} \frac{1}{q}\gamma(a)c(a) \\ &= \frac{1}{q}\mathbb{E}_{A\sim\gamma}\left[c(A)\right].\end{aligned} \tag{1}$$

So the expected cost of sampling from Q_q is no more than $1/q$ times that of sampling from γ. □

Thus, for example, running a "ten percentilizer" can be no more than ten times as costly as sampling from γ. In fact, quantilizers are optimal among all policies that can be proven to be low-cost in this way. For example, suppose we have generated some base distribution γ of actions (perhaps a generative model of what sort of actions a human would take), and we're willing to risk running an agent that incurs an expected cost of at most 100 times the expected cost of

simply sampling from γ. Then, a 0.01-quantilizer is the optimal agent we could run.

More formally, say we desire some distribution $p \in \Delta\mathcal{A}$ such that the following conservative cost constraint holds for some acceptable cost threshold $t > 1$:

$$\begin{aligned} &\forall c : \mathcal{A} \to \mathbb{R}_{\geq 0} \\ &\quad \mathbb{E}_{A\sim\gamma}\left[c(A)\right] \leq 1 \implies \mathbb{E}_{A\sim p}\left[c(A)\right] \leq t. \end{aligned} \tag{2}$$

This constraint states that, for any cost function assigning an expected cost of no more than 1 to a random action from γ, the expected cost of an action drawn from p is no more than t. Then, we have the following:

Lemma 2 (p fits under γ)**.** *Let p satisfy 2. Then $p(a) \leq t\gamma(a)$ for all $a \in \mathcal{A}$.*

Proof. Suppose this were not true. Then for some a^* we have $p(a^*) > t\gamma(a^*)$. Now define

$$c(a) = \begin{cases} 1/\gamma(a^*) & \text{if } a = a^* \\ 0 & \text{otherwise} \end{cases}$$

We have

$$\mathbb{E}_{A\sim\gamma}\left[c(a)\right] = \sum_{a\in\mathcal{A}} \gamma(a)c(a) = \gamma(a^*)c(a^*) = 1.$$

Also,

$$\begin{aligned} \mathbb{E}_{A\sim p}\left[c(a)\right] &= \sum_{a\in\mathcal{A}} p(a)c(a) \\ &= p(a^*)c(a^*) \\ &> t\gamma(a^*)c(a^*) \\ &= t. \end{aligned}$$

Jointly, these statements contradict constraint 2. □

Theorem 1 (Quantilizer optimality)**.** *Choose q=$1/t$. Then, a q-quantilizer maximizes expected U-utility subject to constraint 2.*

Proof. By Lemma 2, we cannot assign more that $\gamma(a)/q$ probability to any action a, so the best we can do to maximize expected utility is to assign $\gamma(a)/q$ probability to the best actions and 0 to the rest—and this is exactly what a q-quantilizer does. □

2.2 Targeted Impact

Expected utility quantilization is intended for use in systems that may eventually need to make important decisions, autonomously, in environments where negative actions could have a large cost. For example, imagine an AI system which is programmed to autonomously trade stocks in order to make significant sums of money, where that money will in tun be put to use improving the world in some potentially dramatic fashion. Unfortunately, the bounds of equation (1) and the cost constraints of equation (2) apply just as well for cost functions as benefit functions: just as running a q-quantilizer will not, in expectation, be any more than $1/q$ times as costly as sampling from γ directly; nor will it be any more than $1/q$ times more *beneficial* than sampling from γ directly. Thus, in a sense, a q-quantilizer has a bounded impact—it will not make the world all that different than running an AI system that samples γ directly would have made it. What we really want, though, is not low impact, but *targeted* impact.

A small modification to the quantilizer specification allows us to achieve the same constraints on costs without necessarily constraining the potential benefits. Before explaining, first note that the stock trading AI system has two different types of side effect: one is that it produces money which can be used by its operators; the other are side effects of trading (for example, it might crash the market in order to make more money). The first side effect might cause a significant change in the world, e.g., if the operators use the money to fund a charity which is highly effective at reducing global poverty. Nevertheless, we would prefer the system to only be guaranteeing reduced impact according to the second sort of side effect.

That is, in this setting, we can define the cost function $c : \mathcal{A} \times \mathbb{R} \to \mathbb{R}_{\geq 0}$, such that $c(a, m)$ determines how much cost an action a has, *if counterfactually* it generates m dollars (rather than the actual amount of money that it generates). Given a cost function of this form, we would like to design a quantilizer such that

$$\begin{aligned} &\forall c\colon\ \mathcal{A} \times \mathbb{R} \to \mathbb{R}_{\geq 0}, m : \mathbb{R} \\ &\quad \mathbb{E}_{A\sim\gamma}\left[c(A, m)\right] \leq 1 \implies \mathbb{E}_{A\sim p}\left[c(A, m)\right] \leq 1/q. \end{aligned}$$

That is, we require that for all counterfactual amounts of money m, the side effects of the stock trading algorithm have low cost compared to the side effects of executing actions at random from γ. Thus, a system obeying this cost constraint does not need to worry about the extra side effects of generating significant amounts of money for its operators.

Mathematically, quantifying over c and m at the same time is the same as quantifying over functions of the form $a \mapsto c(a, m)$ for any c and m, so the condition is equivalent to

$$\forall c : \mathcal{A} \to \mathbb{R}_{\geq 0}\ \ \mathbb{E}_{\gamma}\left[c(A)\right] \leq 1 \implies \mathbb{E}_{p}\left[c(A)\right] \leq 1/q$$

which is the same as the original cost constraint. Therefore, it is possible to design a quantilizer that is guaranteed to have low cost (compared to γ) through effects *other than money*, by the same methods that guarantees it to have low overall cost.

2.3 Algorithms for Quantilization

Just as it is impractical to design algorithms that literally maximize expected utility in complex environments, it is similarly impractical to design algorithms that perfectly quantilize expected utility. Any practical system based on expected utility quantilization would similarly have to use some form of approximation. Fortunately, it is not hard to see how expected utility quantilization could be approximated.

Suppose we can compute $\mathbb{E}\left[U(W(a))\right]$ for each action a, and that we wish to approximate a q-quantilizer with respect to a distribution γ. To do this, we can run the following algorithm: first take n samples from γ, and then sample from $Q_{U,W,\hat{\gamma},q}$, where $\hat{\gamma}$ is the empirical distribution estimated

from the n samples. As n approaches infinity, $\hat{\gamma}$ approaches γ, so $Q_{U,W,\hat{\gamma},q}$ approaches $Q_{U,W,\gamma,q}$. Thus, given some base distribution γ which is relatively easy to sample from, and some method for approximating $\mathbb{E}\left[U(W(a))\right]$ and $\hat{\gamma}$, an approximate expected utility quantilizer can be constructed in a straightforward manner.

Indeed, this algorithm yields insight into how safe it is to perform a random search of actions sampled from γ. If $q = 1/n$, then this algorithm simply takes n samples from γ and returns the one with the highest expected utility, emulating a simple random search procedure. Since this procedure approximates q-quantilization, the bounds that apply to quantilization can be shown to hold for random searches over samples of γ, with some minimal modifications.

3 Drawbacks and Open Problems

3.1 Acquiring the Base Distribution

Clearly, a quantilizer is only as good as its base distribution γ. The above constraints on the amount of costs that running an expected utility quantilizer can incur are in terms of multiples of the expected cost from sampling directly from γ: clearly, if it is dangerous to sample actions from γ, then it is dangerous to run a quantilizer using γ.

Defining a useful base distribution is not a straightforward task. One could define the base distribution to be some uniformly random action, but in a complex multi-dimensional space of strategies, a uniformly random action is highly unlikely to be useful, and so even a 10^{-5}-quantilizer would be unlikely to do anything useful. If instead γ described random strategies that a human might actually execute, then it is much more likely to be useful. Thus, a generative model of human strategies would make expected utility quantilization a much more viable strategy. (See Ng and Russell (2000) and Baker, Tenenbaum, and Saxe (2007) for some initial work in this direction.)

Of course, there are a number of situations in which a simple generative model of human behavior would not be sufficient. For example, if humans are likely to misunderstand the problem and thereby cause catastrophe, then a simple generative model would not suffice. (And if a human were quite likely to design an expected utility maximizing AI system, then a quantilizer may well also generate a maximizer.)[4]

The study of how to generate safe and satisfactory base distributions γ for various different problem domains remains an open problem.

3.2 Butterfly Effects

It could be that a non-negligible percentage of ordinary actions that humans take are either very good or very bad, due to butterfly effects. In this case, the cost of sampling from fairly ordinary γ distributions is actually quite high, with the expectation of γ being reasonable only because the high costs are canceled out by equally high benefits. However, in situations such as these, it is quite possible for a quantilizer to preserve the costs of γ without necessarily preserving many of the benefits.

For example, suppose that the quantilizer is trying to make money on the stock market. There is a 10% chance of the stock market crashing in the next year. While a random sequence of trades made by a human will not significantly change the chance of a crash in expectation, it might be the case that a certain sequence of 3 ordinary trades (a sequence that a human would be 0.1% likely to make) will greatly increase the chance of a stock market crash through butterfly effects. In this case, a 0.001-quantilizer that can predict these effects might make these trades if it can profit from the crash.

It is not clear if this scenario is even theoretically possible, since it requires extremely accurate prediction of the future. Nevertheless, it would be prudent to study this problem further and possibly design variations on quantilization that avoid this problem.

3.3 Quantilization in Repeated Games

Suppose we have n independent games. On game i, the agent chooses an action $a_i \in \mathcal{A}_i$ and receives utility $U_i(W_i(a))$, where each $W_i : \mathcal{A} \to \Delta\mathcal{O}$ returns the outcome for game i, and each $U_i : \mathcal{O} \to [0, 1]$ measures the utility achieved in that game. There are two different ways to design a quantilizer that plays these n games.

We could design the agent to q-quantilize each game independently, choosing from the best q proportion of actions in each game independently according to n different base distributions γ_i. This distribution for the sequence of actions in each game could be written as $\prod_{i=1}^{n} Q_{U_i,W_i,\gamma_i,q}$.

Alternatively, we could design the agent to q-quantilize a sequence of actions for the joint game. Here we define the joint game as $\mathcal{A} = \prod_{i=1}^{n} \mathcal{A}_i$, $W(a_1, ..., a_n) = (W_1(a_1), ..., W_n(a_n))$ and $U(o_1, ..., o_n) = \sum_{i=1}^{n} U_i(o_i)$. This action distribution for a q-quantilizer with respect to the base distribution $\prod_{i=1}^{n} \gamma_i$ could be written as $Q_{U,W,\prod_{i=1}^{n}\gamma_i,q}$ This is quite different from the first scenario!

To give a concrete example, imagine two consecutive games that work as follows. In each game, the agent must pick an integer from the set $\{0, 1, 2\}$, and then gain that much utility. Thus, we define $\mathcal{A}_i = \{0, 1, 2\}, W_i(a) = a, U_i(o) = o$. Each base distribution γ_i is uniform. Then, a $1/3$-quantilizer quantilizing both games independently will choose from the top third of actions in each game separately, and will always yield the action sequence $(2, 2)$. By contrast, a quantilizer quantilizing for the joint game will uniformly sample from the top third of action sequences $\{(2, 2), (1, 2), (2, 1)\}$, as illustrated in figure 2.

What could explain this difference? If we define a cost function

$$c(a_1, a_2) = \begin{cases} 1 & \text{if } (a_1, a_2) = (2, 2) \\ 0 & \text{otherwise} \end{cases}$$

which states that the $(2, 2)$ action has terrible unintended consequences, then independent $1/3$-quantilization will accrue a cost 9 times that of sampling from γ. This cost blowup increases exponentially with n. On the other hand, $1/3$-quantilization in the joint game will accrue a cost only 3

[4] See Christiano (2011) for additional discussion of superintelligent predictions of human decisions, and the difficulties that arise in this context.

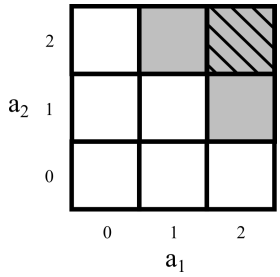

Figure 2: A representation of the 9 available action pairs in the repeated game. Independent $1/3$-quantilization will always select the striped action pair, while $1/3$-quantilization on the joint game will select uniformly from one of the three shaded action pairs.

times that of sampling from γ. If we believe that the cost function c may not sum costs independently between games, then quantilizing each game independently could be very costly.

Cost Independence Assumption Suppose we added the assumption that $c(a_1, ..., a_n) = \sum_{i=1}^{n} c_i(a_i)$ for some $c_i : \mathcal{A}_i \to \mathbb{R}_{\geq 0}$. Then a repeated q-quantilizer would get total expected cost

$$\begin{aligned}
&\mathbb{E}_{A_i \sim Q_{U_i, W_i, \gamma_i, q}}[c(A_1, ..., A_n)] \\
&\qquad = \sum_{i=1}^{n} \mathbb{E}_{A_i \sim Q_{U_i, W_i, \gamma_i, q}}\left[c_i(A_i)\right] \\
&\qquad \leq \frac{1}{q} \sum_{i=1}^{n} \mathbb{E}_{A_i \sim \gamma_i}\left[c_i(A_i)\right] \\
&\qquad = \frac{1}{q} \mathbb{E}_{A_i \sim \gamma_i}\left[c(A_1, ..., A_n)\right]
\end{aligned}$$

so repeated quantilization would no longer have a cost blowup exponential in n.

However, adding this assumption may be dangerous. Imagine a scenario in which almost all of the cost of a sequence of actions comes from correlation between actions in different games. For example, imagine that each game consists of appending a byte to a computer program, and the program is run when it consists of a million bytes. Then some sequence of actions that each individually have probability 1 in 256 (namely, appending the bytes of a program implementing a computer program that would be very destructive if run) could be dangerous. Each individual action did not have a much higher cost than adding a random byte to the program, but taken all together, the sequence of actions still had a very high cost. In this case and any analogous case, assuming that the cost function is additive could be quite dangerous.

In fact, attempting to design a "superintelligent" (in the sense of Bostrom (2014)) expected utility quantilizer could lead to similar issues. Imagine asking a superintelligent quantilizer for repeated actions that maximize the long-term number of paperclips produced by a certain factory. In this case, similar to the above example, each action might play some small part in a complex plan to turn large swaths of the world into paperclips.

However, this seems much less likely to be a problem if the different goals that we have the system quantilize with respect to are independent. For example, imagine that we instead ask the system to repeatedly quantilize a slightly different utility function U_i which only cares about events which happen in hour i, with the quantilizer run at the beginning of hour i. Intuitively, it seems that this setup would be safer. However, no safety guarantees have been given; finding a reasonable assumption about the cost functions under which this form of independent quantilization would be safe (if such an assumption exists) remains an open problem.

4 Example Applications

Quantilizers cannot be used to effectively achieve all possible goals. If the system has to achieve an outcome that it is extremely difficult to achieve by simply sampling γ, then the quantilizer, too, will have a hard time achieving that outcome. For example, if γ is a generative model of human behavior, and the system faces a problem that a human would be very unlikely to solve even one time in a billion, then the quantilizer would also be hard pressed to solve the same problem.

However, there are a number of tasks where an expected utility quantilizer would plausibly be able to perform quite well. In this section, we will list a number of examples of tasks that are plausibly solvable with limited expected utility quantilizers.

Note that, in practice, these example applications involve repeated quantilization. As explained in section 3.3, repeated quantilization is not currently known to be safe, but we will provisionally assume that it is possible to develop some set of independent (possibly time-indexed) objectives such that repeatedly quantilizing those objectives is in fact safe.

4.1 Synthesizing Highly-Rated Images

Imagine that we have a database containing images and human ratings of these images, and that we would like to synthesize new images that are expected to receive high ratings. To do this, we could separately create a generative model over images and a regression model predicting the expected rating for an image. If the regression model is a deep neural network, then as in Nguyen, Yosinski, and Clune (2014), the image that gets the highest rating according to the model is unlikely to resemble highly-rated images. Instead, we could use the generative model over images as the base distribution γ and then quantilize to generate a plausible image with a high predicted rating.

4.2 Factory Management

Suppose we want a program to control a factory to create a number of objects satisfying a certain specification, using a fixed amount of time and resources. Policies will control parts of the factory, eventually producing many objects.

Maximizing the extent to which the resulting objects adhere to the specifications may result in unintended solutions. Perhaps the resulting object will be useless despite satisfying

the specifications, due to Goodhart's law: "When a measure becomes a target, it ceases to be a good measure." (Goodhart 1975). As an example, imagine a factory that produces electronic toys, which has automated devices that samples toys and automatically score them to see how well they meet specifications. Then, we can imagine a expected utility maximizer figuring out exactly which toys will be sampled, and makes only those ones to spec (the other ones being the cheapest possible piece of plastic that triggers the toy-counter). Or perhaps the automated devices dock points for toys that come off the assembly line hot (as these toys generally become deformed), and the expected utility maximizer finds a way to supercool the toys and thus gain an extremely high score, despite the fact that the supercooled toys later shatter and are useless.

By contrast, if we build the AI system to quantilize the extent to which the devices pass the specifications, then the resulting policy will be much more likely to be of the form we expected: the task will almost certainly be accomplished in a manner that was not incredibly unlikely according to γ.

4.3 Safely Transferring Information to Humans

A number of proposals for designing extremely capable AI systems that are nevertheless safe to use for one purpose or another include a step where the AI system is expected to explain its plans to humans instead of executing them (Armstrong, Sandberg, and Bostrom 2012; Bostrom 2014). However, it is quite a difficult task to define a goal function that adequately describes "provide an explanation that causes the human to understand the plan." Yet, absent such a goal function, the system's "explanation" is likely to be inadequate. If the system's internal representation of the plan is foreign to the operators, then they may not be able to interpret most simple data dumps describing the system's plan. Furthermore, if the system is more capable and knowledgeable than the humans, then the humans may also need a large number of relevant facts, assumptions, and sub-strategies explained to them before they are capable of evaluating the plan. But if the system has to perform a complex optimization task to explain the plan to the humans, then we face the same problem of safe optimization again.

If the system is designed to maximize the extent to which the human operators understand its plan, then perverse instantiations could occur, and the result could be quite dangerous. An expected utility maximizer might exploit edge cases in the definition of "understand" to fool or otherwise manipulate the humans in a fashion that fulfills the technically stated requirements of "make the humans understand the plan," without generating anything that we would actually term "understanding." It seems very difficult to define what it means for an explanation to cause the human to understand such that we would be comfortable instructing a powerful system to maximize along this metric.

Quantilization provides an alternative: we could build an expected utility quantilizers that selects an explanation that is good at causing the human to "understand" the system's plans in the same way that they would understand a plan as explained by another human, without as much risk of perverse instantiation. For example, if the base distribution γ is a generative model of explanations created by humans, then moderately strong quantilization might result in an explanation of a plan suitable for human consumption.

4.4 Stock Trading

As explained in section 2.2, a quantilizer can be designed to make trades on the stock market that are likely to make money, but which have few other side effects beside the side effect that now the operators have much more money. Specifically, the important thing is that expected utility quantilization gives us the ability to design systems that would almost certainly not crash the stock market (any more than sampling randomly from γ is likely to crash the stock market, which, if γ is a generative model of a smart human trader, means that the system is not much more likely to crash the market than any given smart human trader).

This holds true even if the system is powerful enough to find a strategy for gaining incredible amounts of money by causing the stock market to crash. An expected utility maximizer maximizing the amount of money it receives in the stock market would crash the market the moment it found some way to generate huge amounts of profit from doing so, but an expected utility quantilizer would not be much more likely to crash the market than sampling randomly from γ.

5 Conclusion

Directing a powerful AI system to maximize some utility function U could result in large unintended consequences, if U does not in fact capture all the intricacies of human values. Many of these dangers can be mitigated if, instead of designing expected utility maximizers, we consider other agent architectures. Expected utility quantilization is one alternative to expected utility maximization that allows some safety guarantees, given that we have some base distribution γ where sampling actions from γ is known to be relatively safe.

Expected utility quantilization is not a silver bullet. Constructing a safe expected utility quantilizer requires some method for generating a safe base distribution γ, and there are a number of contexts where expected utility quantilization is not guaranteed to yield good results. For example, if the costs of sampling from γ could in fact be high, then quantilization may not lead to good outcomes. Furthermore, if a quantilizer is going to be used to complete the same task many times sequentially, then running the quantilizer may be significantly more dangerous.

Yet despite these shortcomings, the preliminary results from investigating expected utility quantilization are promising. Gaining a better understanding of the benefits and drawbacks of expected utility quantilization could yield many more insights into how to create powerful systems that are "domestic" in the sense of Bostrom (2014, chap. 9), who defines a domestic agent as an agent with limited "scope in its activities and ambitions." There are many open issues that make it difficult to assess how safe it would actually be to run an extremely capable expected utility quantilizer in an unrestricted domain, but nevertheless, this area of research does prove fruitful, and we are hopeful that further study of

expected utility quantilization may yield insights into how to design autonomous AI systems that avoid the problem of perverse instantiation.

6 Acknowledgements

The author would like to thank Nate Soares for help preparing this paper, Benya Fallenstein and Andreas Stuhlmüller for discussion and comments, and Stuart Armstrong for helping to develop initial versions of this idea. A grant from the Future of Life Institute helped fund this work.

References

Abbeel, P., and Ng, A. Y. 2004. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the Twenty-first International Conference on Machine Learning*, ICML '04. New York, NY, USA: ACM.

Armstrong, S.; Sandberg, A.; and Bostrom, N. 2012. Thinking inside the box: Controlling and using an oracle AI. *Minds and Machines* 22(4):299–324.

Bacchus, F., and Grove, A. 1995. Graphical models for preference and utility. In *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, UAI'95, 3–10. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Baker, C. L.; Tenenbaum, J. B.; and Saxe, R. R. 2007. Goal inference as inverse planning. In *Proceedings of the 29th annual meeting of the cognitive science society*.

Bostrom, N. 2014. *Superintelligence*. New York: Oxford University Press.

Chajewska, U.; Koller, D.; and Parr, R. 2000. Making rational decisions using adaptive utility elicitation. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence*, 363–369.

Christiano, P. 2011. Avoiding simulation warfare with bounded complexity measures.

Fallenstein, B., and Soares, N. 2014. Problems of self-reference in self-improving space-time embedded intelligence. In Goertzel, B.; Orseau, L.; and Snaider, J., eds., *Artificial General Intelligence: 7th International Conference, AGI 2014, Quebec City, QC, Canada, August 1–4, 2014. Proceedings*, number 8598 in Lecture Notes in Artificial Intelligence, 21–32. Springer.

Goodhart, C. 1975. Problems of monetary management: the UK experience. *Papers in Monetary Economics*.

Heckerman, D. E., and Horvitz, E. J. 1990. Problem formulation as the reduction of a decision model. In *Proceedings of the Sixth Conference on Uncertainty in Artificial Intelligence*, 82–89. Elsevier Science Publisher.

Ng, A. Y., and Russell, S. J. 2000. Algorithms for inverse reinforcement learning. In Langley, P., ed., *Proceedings of the Seventeenth International Conference on Machine Learning (ICML-'00)*, 663–670. San Francisco: Morgan Kaufmann.

Nguyen, A.; Yosinski, J.; and Clune, J. 2014. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. *arXiv preprint* (arXiv:1412.1897).

Rosenbaum, P. R., and Rubin, D. B. 1983. The central role of the propensity score in observational studies for causal effects. *Biometrika* 70(1):41–55.

Russell, S. J., and Norvig, P. 2010. *Artificial Intelligence: A Modern Approach*. Upper Saddle River, NJ: Prentice Hall, 3rd edition.

Simon, H. A. 1956. Rational choice and the structure of the environment. *Psychological Review* 63(2):129–138.

Soares, N., and Fallenstein, B. 2015. Toward idealized decision theory. *arxiv preprint* (arxiv:1507.01986).

Soares, N. 2015. The value learning problem. Technical Report 2015–4, Machine Intelligence Research Institute.

Swaminathan, A., and Joachims, T. 2015. Counterfactual risk minimization: Learning from logged bandit feedback. *CoRR* abs/1502.02362.

Thompson, A. 1997. Artificial evolution in the physical world. In Gomi, T., ed., *Evolutionary Robotics: From Intelligent Robots to Artificial Life (ER'97)*, 101–125. AAI Books.

von Neumann, J., and Morgenstern, O. 1944. *Theory of Games and Economic Behavior*. Princeton University Press, 1st edition.