# **How to Specify** *Aligned* **Reinforcement Learning Problems**

Brad Knox



## **Outline**

- 1. Aligned reward functions
- 2. Finding misalignment in a reward function
- 3. Misalignment by shaping
- 4. Misalignment by trial-and-error design
- 5. Misalignment by discounting
- 6. Designing aligned reward functions
- 7. What should we work towards?

# **Aligned reward functions**



## **Gridworld**



### States: cell location

### Actions: move in each cardinal direction





State: board configuration

Action: a legal move





Observation: joint positions, joint velocities, and a camera image of its task area

Action: acceleration on each joint



# **Reinforcement learning**



## **Reinforcement learning**

## **Gridworld** State: cell location



Action: move in each cardinal direction

Reward: -1 per time step / action



## **Reinforcement learning**



State: board configuration

Action: a legal move

Reward: 1 upon winning, 0 otherwise (assume no stalemates)



## **REWARD AND RETURN**

 $T$  $G(\tau) = \sum \gamma^t R(s_t, a_t, s_{t+1})$  $t=0$ return reward

## **REWARD AND RETURN**

## **Field**

-

-

reinf. learning motion planning control theory evolutionary algs. utility theory optimization



\* "objective function" more precisely refers to the expectation of  $G(\tau)$ 



## **EXPECTED RETURN**

# $J(\pi)=E_{\pi}[G(\tau)]$

(Implicit in the expectation is the distribution over start states and state transitions.)



## **EXPECTED RETURN**

$$
J(\pi)=E_{\pi}[G(\tau)]
$$

(Implicit in the expectation is the distribution over start states and state transitions.)

**A more precise characterization of RL:** attempt to find a behavior policy π that maximizes **expected return.**

## **Rewards vs goals**



## **Main alternatives to RL problems**

- **● Learning from demonstrations** (imitation learning)
- **● Learning from preferences** (RLHF, RLAIF, DPO, CPL, etc.)

*Learning an intermediate reward function and doing RL on it is not the only way to these methods.*

## **BACKGROUND ON REWARD**

RL oversimplified: a set of problems and corresponding algorithmic solutions, in which *experience in a task is used to improve an agent's behavior such that it gets more reward*.

More specifically, most RL problems focus on increasing the *expectation* of G(τ), the utility of a trajectory:

$$
G(\tau) = \sum_{t=1}^{(T-1)} R(s_t, a_t, s_{t+1})
$$

(Assumes undiscounted/episodic setting and an unstated distribution over starting states)

## **An aligned reward function**

A **reward function** creates a preference ordering over possible trajectories (by  $G(\tau)$ ) and probability distributions over trajectories. These trajectories can simplified to only the outcomes that matter (e.g., winning/losing or time until reaching a goal.) We assume **humans** also have such an ordering.

**A perfectly aligned reward function creates an ordering over outcome distributions that matches that of the human stakeholder.**

## **An aligned reward function**



**A perfectly aligned reward function creates an ordering over outcome distributions that matches that of the human stakeholder.**

## **An aligned reward function**



**A perfectly aligned reward function creates an ordering over outcome distributions that matches that of the human stakeholder.**

## **An aligned reward function?**

### Human stakeholder **Expected** return



## **An aligned reward function?**

### Human stakeholder **Expected** return



## **An aligned reward function?**

## Human stakeholder **Expected** return



## **Gridworld** State: cell location



Action: move in a cardinal direction

The agent's purpose is to reach the goal in the minimum mean time from any state.

**Design a reward function** by designating what the reward should be for each action from each cell (state).

*There is no discounting and the goal state is terminal (or transitions to absorbing state, if you prefer).*



### **Gridworld An aligned reward function**

 *-1 reward until goal*

The return is the time to goal  $(* -1)$ . The **expected return** is the mean time to goal  $(* -1)$ . So the task is to minimize the mean time to goal.



### **Gridworld A misaligned reward function**

 *1 reward for an action towards the goal 0 reward otherwise*

The return is the number of goal-approaching actions. The **expected return** is the mean number of goal-approaching actions. So the task is to maximize the mean number of goal-approaching actions.



### **Gridworld A misaligned reward function**

 *1 reward for an action towards the goal 0 reward otherwise*

The return is the number of goal-approaching actions. The **expected return** is the mean number of goal-approaching actions. So the task is to maximize the mean number of goal-approaching actions.

Will an agent that's maximizing its expected return terminate?



### **Gridworld A misaligned reward function**

 *0 reward for an action towards the goal -1 reward otherwise*

The return is the number of suboptimal actions (\* -1). The **expected return** is the mean number of suboptimal actions (\* -1). So the task is to minimize the mean number of suboptimal actions.



### **Gridworld A misaligned reward function**

 *0 reward for an action towards the goal -1 reward otherwise*

The return is the number of suboptimal actions (\* -1). The **expected return** is the mean number of suboptimal actions (\* -1). So the task is to minimize the mean number of suboptimal actions.

What if the goal is moved?

What if you misunderstood which actions go to the goal?

# **AI safety terminology**

*Precise definitions couldn't be found, so my versions:*

**outer alignment** - the problem given to an AI optimizer to solve is aligned

● regardless of whether the resultant solution is aligned in practice

**outcome-based learning** - optimizing decisions based on future rewards or goals

# **Is RL unfixably unsafe?**



### **POLICY FORUM**

**ARTIFICIAL INTELLIGENCE** 

## Regulating advanced artificial agents

Governance frameworks should address the prospect of AI systems that cannot be safely tested

By Michael K. Cohen<sup>1,2</sup>, Noam Kolt<sup>3,4</sup> Yoshua Bengio<sup>5,6</sup>, Gillian K. Hadfield<sup>2,3,4,7</sup>, Stuart Russell<sup>1,2</sup>

echnical experts and policy-makers have increasingly emphasized the need to address extinction risk from

under control is also reflected in President. Biden's 2023 executive order that introbe controlle duces reporting requirements for AI that achieve com could "eva[de] human control or oversight rewards ap through means of deception or obfuscacapable RL tion"  $(3)$ . Building on these efforts, now is rewards, wh the time for governments to develop regul to secure me

So long

"Giving an advanced AI system the objective to maximize its reward (LTPAs)..." leads to concerns that include reward tampering, removing humans as obstacles to reward, and power seeking.

"both safety and validity cannot be ensured when testing sufficiently capable LTPAs"

"**Developers should not be permitted to build sufficiently capable LTPAs, and the resources required to build them should be subject to stringent controls.**"

## **Is RL unfixably unsafe?**



### **POLICY FORUM**

**ARTIFICIAL INTELLIGENCE** 

## Regulating advanced artificial agents

Governance frameworks should address the prospect of AI systems that cannot be safely tested

By Michael K. Cohen<sup>1,2</sup>, Noam Kolt<sup>3,4</sup> Yoshua Bengio<sup>5,6</sup>, Gillian K. Hadfield<sup>2,3,4,7</sup>, Stuart Russell<sup>1,2</sup>

echnical experts and policy-makers have increasingly emphasized the need to address extinction risk from

under control is also reflected in President. Biden's 2023 executive order that introduces reporting requirements for AI that could "eva[de] human control or oversight through means of deception or obfuscation" (3). Building on these efforts, now is the time for governments to develop regul

So long be controlle achieve com rewards ap capable RL rewards, wh to secure me

"If dangerously capable LTPAs are at some point permitted to be developed, rigorous technical and regulatory work would need to be done first..."

**This talk covers such work.**

## **Is RL impactful?**



Observation: prompt + any previous text

Action: n-token response

Reward: ???



## **Is RL impactful?**



Observation: prompt + any previous text

Action: n-token response

Reward: ???

*Here, demonstrations and preferences are used. The "reward function" from RLHF is not a reward function in they way we normally use*  the concept.



# **Is RL impactful?**

RL has often had a stigma of **not yet working well for important problems.** It has had some large successes though. A few:

- Games: Go. Chess, Poker, and Starcraft
- Data center cooling
- RL as search (LLM fine-tuning and AlphaFold)

Learning from long-horizon reward is **harder to wield than learning from demonstrations and preferences.** 

**Nonetheless, a well-formulated RL problem has the potential to lead to performance far beyond what humans can demonstrate or identify through preferences.**

## **The set of optimal policies is invariant to rescaling of the reward function.**
### **The set of optimal policies is invariant to shifts of the reward function if...**

from each state, all possible trajectories have the same length

Includes continuing and finite horizon tasks.

Does not include typical episodic tasks, such as those with goal or failure states.

### **A change in perspective**

### **Reward from the perspective of an RL algorithm** ← the familiar perspective

An agent conducting policy improvement continually searches for policies that get higher mean return.

We start zoomed in and zoom out from there:

- Agent is a pursuer of reward.  $\left(\frac{?}{\cdot}\right)$
- Agent estimates expected return from the reward it has experienced.
- Agent identifies actions (more precisely, changes in policy) that will increase estimated expected return.

### **A change in perspective**

### **Reward from the perspective of an RL algorithm** ← the familiar perspective

An agent conducting policy improvement continually searches for policies that get higher mean return.

We start zoomed in and zoom out from there:

- Agent is a pursuer of reward.  $\left(\frac{?}{\cdot}\right)$
- Agent estimates expected return from the reward it has experienced.
- Agent identifies actions (more precisely, changes in policy) that will increase estimated expected return.

### **A change in perspective**

### **A reward-centric perspective of policies**

The choice of the reward function and discounting create an ordering over policies—given a start-state distribution—via their expected returns and over full trajectories.

Each change in a reward function may rearrange this ordering by changing each policy's expected return.

Learning is not a consideration.

Ordering by expected return:



# **Consider designing for interpretable return.**

Two highly common reward functions have interpretable return.

*reward function: -1 reward until goal* The return is the time to goal  $(* -1)$ . The **expected return** is the mean time to goal (\* -1). So the task is to minimize the mean time to goal.



*reward function: 0 upon losing, 1 upon winning, and 0 otherwise* The return is a binary indicator of winning. The **expected return** is the probability of winning. So the task is to maximize the probability of winning.



# **Finding Misalignment in a Reward Function**

### **FIND MISMATCHES IN PREFERENCE ORDERINGS** (The most powerful method I'm aware of.)

**If all human stakeholders agree that trajectory**  $\mathbf{r}^{}_{\mathsf{A}}$  **is preferable to**  $\mathbf{r}^{}_{\mathsf{B}}$ **(i.e.,**  $\tau$ <sub>A</sub>  $> \tau$ <sub>B</sub>), then return of  $\tau$ <sub>A</sub>  $>$  return of  $\tau$ <sub>B</sub> should hold.



W. Bradley Knox, Alessandro Allievi, Holger Banzhaf, Felix Schmitt, and [Peter Stone](http://www.cs.utexas.edu/~pstone). **Reward (Mis)design for Autonomous Driving**. AIJ 2023.

















### End-to-End Race Driving with Deep Reinforcement Learning

Maximilian Jaritz<sup>1,2</sup>, Raoul de Charette<sup>3</sup>, Marin Toromanoff<sup>2</sup>, Etienne Perot<sup>2</sup> and Fawzi Nashashibi<sup>1</sup>

Abstract—We present research using the latest relative<br>coming algorithm for end-to-end driving without any mediated<br>preception (object recognition, scene understanding). The newly<br>preposed reviewd and kaming strategies lea uro ano surning strategio suo togener o<br>sose and more robust driving using only RGI a forward facine camera. An As Published reward functions! We propose a method (fig. II) benefiting from roomt agent lear **for autonomous driving** hairpin bends with few crashes (drawn in yellow). whereas. London to exclude the detection in Eq. 10.1 and the of three forests are presented in [19] with a present the detection of real video is shown in [2] we see point. Similar results were presented in [19] with a lo

> 19 papers reviewed ive experiments on CARLA driving benchmark de-CIRL substantially outperforms all previous methods in terms of the percentage of successfully completed episodes on a variety of goal-directed driving tasks. We also show its superior generalization capability in unseen environments. To

**CIRL: Controllable Imitative Reinforcement Learning** 

for Vision-based Self-driving

our knowledge, this is the first successful case of the learned driving policy by reinforcement learning in the high-fidelity simulator, which performs better than supervised imitation learning.

Keywords: Imitative reinforcement learning, Autonomous driving

### 1 Introduction

S

arXi

Autonomous urban driving is a long-studied and still under-explored task [1,2] par ticularly in the crowded urban environments  $[3]$ . A desirable system is required to be canable of solving all visual perception tasks (e.g. object and lane localization, drivable paths) and determining long-term driving strategies, referred as "driving policy" Although visual perception tasks have been well studied by resorting to supervised learning on large-scale datasets [45], simplistic driving policies by manually designed rules in the modular pipeline is far from sufficient for handling diverse real-world cases is discussed in [67]. Learning a optimal driving policy that mimics human drivers is less explored but key to navigate in complex environments that requires understandine of multi-agent dynamics, prescriptive traffic rule, negotiation skills for taking left

### Deep Distributional Reinforcement Learning Based High-Level Driving Policy Determination

Kyushik Min<sup>®</sup>, Hayoung Kim®, and Kunsoo Huh®, Member, IEEE



In this study, we train the supervisor based on deep rein forcement learning and the raw sensor output so that vehicle<br>forcement learning and the raw sensor output so that vehicle<br>agent rans as fast as possible with minimizing unnecessary lane **(m) Excel** a consistent in the form of the three interactions are consistent in the constraints of the second of the s

EEE TRANSACTIONS ON INTELLIGENT VEHICLES, VOL. 4, NO. 3, SEPTEMBER 2015

processors person masses are one was used to a lifted a lifted and the state of the state of

**Manufactures (September 1978)**<br> **Excellent Constitution (September 1978)**<br> **Excellent Constitution (September 1978)**<br>
2000 **Constitution (September 1978)**<br>
2000 **Constitution (September 1978)**<br>
2000 **Constitution (Septemb** the related works on learning based approaches for autonomous<br>driving are summarized. In Section III, the Markov deciso a senior want may engine any. iens.org<br>usable: 10.1109TPC2019.2929457 sion process is briefly explained as the basis of reinforcement

 $2179\,4858\,$  O  $2019\,$  EEE. Personal use is permitted, but republication/reductionism requires LEEE per See http://www.ieec.org/publications\_madach/publications/rights/index httel for more informational results

Ized licensed use limited to: University of Texas at Austin. Covenlanded on January 27,0021 at 20:52:00 UTC from IEEE Xplore. Farmicrians appl

APPEARED IN<br>Proc. Robotics: Science & Systems (RSS), 2019

Abstract—Even<br>out\_commerciality<br>projects as highway<br>and can only<br>and can only<br>projects and 1<br>projects and 1<br>and 1<br>and 1<br>and 1

1. INTRODUCTION

### LeTS-Drive: Driving in a Crowd by Learning from Tree Search

### Panpan Cai, Yuanfu Luo, Asoem Saxena, David Hsu, Woe Sun Lee School of Computing, National University of Singapore, 117417 Singapo

nci—Autonomen driving in a crowded customa<br>traffic intersection, is an unselved challenge for<br>het vehicle must contend with a dynamic and bet vehicle must constant with a dynamic and garding<br>the conferences of the vehicle must constant in the format space of<br> $\sim$  10.0 magnitude is the format<br>in the format in the constant in the constant in the constant of t the search. In the outlies peace, the learned past<br>**Andre School and the search. LeTS Drive is**<br>**Andre School and the runline efficiency of and the performance of planning**<br>**Andre School and the search of the search of the** 

The 1: Autonomous driving in a crowd. A robot vehicle drive<br>
1. INTRODUCTION<br>
target provided streets, markets, worlds many moving podentiams. Each pedentian moves to<br>
send the crowded streets, markets, weld the vehicle.<br>

- squares without colliding with any others and make their
- casily fail in these scenarios. The cosinonment is highl
- 

environment dynamics on the belief. Complex problems require neural network that approximates the value iteration algo celine planning: perform a look-ahead search in a belief tree to to provide an initial plan, which is refined by another neural controls a solicy stockts the first action in the rolling and re- network commonse



### **Dynamic Input for Deep Reinforcement Learning** in Autonomous Driving

**End-to-End Model-Free Reinforcement Learning** 

for Urban Driving using Implicit Affordances

Marin Toromanoff<sup>4,2,3</sup>. Emilie Wirbel<sup>2,3</sup>. Exhien Montarde<sup>1</sup> Malin Torontanon - , Entime Wilson - , Patricia Moutanoe<br>Center for Robotics, MINES ParisTech, PSL <sup>3</sup>Valoe Driving Assistance Research <sup>1</sup>Valoe.ai

a na naoveza nastao i mateixa i su:<br><sup>1</sup>name - su reamedini nasi-matri at noh - d'r .........<sup>33</sup>s ann - su reamediva i no - orn

cases do exist but they are currently mostly limited to land where we take the control [1, 34].<br>Deep Reinforcement Learning (DRL) on the other side<br>lets the algorithm learn by itself by providing a reward sig-

nal at each action taken by the agent and thus does not suffer

from distribution relatests. This spaced can be enough took from distribution missantals. This rewards can be space and<br>the choice is not shown in the agent should have done but the space of<br> $\alpha$  and the space of the station taken is locally. The fit<br>all  $\alpha$  is a fit and the spac

than supervised learning to converge, which can load to

ficulties when training large networks with many parameters

ters. Moreover many RL algorithms rely on a replay buffer

ters. Moreover many RL algorithms rely on a replay buffer<br>[21, 25, 12] allowing to kann from past experiments but<br>such buffers can limit the size of the input sued (e.g. the size<br>of the finance). That is why nonnal networ

to solve such complicated tasks as urban driving. Therefore

current DRL approaches to autonomous driving are applied

to simpler cases, e.g. only steering control for lane locating

privileged information as auxiliary losses also coined affec-

dances in some recent papers 12, 311. The idea is to train a network to prodict high level information such as some a network to product right tover information such as seman-<br>tic segmentation maps, distance to center of the lane, traffic<br>light state etc... This prediction can then be used in sev-

eral ways, either by a classic controller as in Sauer et al [31], either as acceliary loss helping to find better features

to the main irritative task loss as in Mehra et al. [23] or also

in a model-based RL approach as in the really recent week

of Pan et al. [26] while also providing some interpretable

comes highed a collision homeon does come-

reason behind a collision between two autoniumou<br>was "fallure to anticipate vehicle intent" [10]. Eve<br>higher order dynamics, the often-unprodictable <mark>behav</mark>

human drivers complicates the use of rule-based algorithms<br>Second, in many cases an agent using TTC is overly custions

creating unsecessary delays. Third, TTC methods assume fall knowledge of their surroundings limiting that ability to

used for the intersection handling, such as insistion learning,<br>online planning, and offline learning. In imitation learning,

the policy is learned from a human driver [11]. however this the policy is learned from a human driver [11]. however this policy does not offer a solution if the agent finds itself in a source that is easy to at the street of the training data. Ordite planeters somewhere the best ac

coissence of an accurate generative model. Offline barring<br>excites the intersection problem, often by using Markey<br>Decision Processes (MDP) in the back-end [13], [14].

are approach to proceed to provide the state of the relationship of the relationsh

ran serenasp in unu surruanangs manag unu seres i handle occlusions. While special behaviors can be coded to<br>address specific cases, this requires an explaner to identify<br>and manage a large possible number of scenarios. These<br>resources methods con irrestigation of machine

feedback on how the decision was taken

Navigating Occluded Intersections with Autonomous Vehicles

using Deep Reinforcement Learning

Texas ( Fig. 1: Using complexy below

David Isele<sup>1,3</sup>, Reza Rahimi<sup>2</sup>, Akansel Cosgur<sup>3</sup>, Kaushik Sabramanian<sup>4</sup> and Kikuo Fujimura

here has a state of the state of the film of the Michigan state of the<br>control width of the state of the state of the state of the state of<br> $\sim$  10  $\mu$  m state of the state of<br>the state of the state of the state of the s

[18] or going as fast as possible in racing games [2] Another dreaduck of DRI should with II is that the Anomer drawback or DML, stated with it, is that it<br>geriften appears as a black box from which it is diffic<br>inderstand how the decision was taken. nderstand how the decision was taken.<br>A promising way to solve both the data efficiency (purticolarly for DRL) and the black box problem is to use

Abstract

Reinforcement Learning (RL) aims at learning an antimal behavior policy from its own experiments and not rais-<br>based control methods. However, there is no RL algorithm

on consider of bouding a top at difficult as when driv

.<br>ing. We present a novel technique, coined implicit affor-<br>dances to effectively leverage RL for arbon driving that insomen, as a gensory secretary and rehidita anong mass<br>challey law keeping, pedentrians and rehidita anothence, and traffic light detection. To our knowledge we are the first<br>to present a naccurated RL agent hasneling mean

by winning the Comera Only track of the CARLA challenge.

Urban driving is probably one of the hardest situations

to solve for autonomous cars, particularly regarding the interaction on intersections with traffic lights, pedestrians

crossine and cars goine on different ressible lanes. Solv-

crossing and cars gestag on different possible lases. Solv-<br>ing this task is still an open problem and it seems compli-<br>cand to handle such diffulny and highly satisfied sinustions<br>with clausic rules-hand approach. This i

Initation learning (IL) [23] aims to reproduce the behav-

is<br>red an expect (a laterata diricur for anomonymus driving by mixele control.<br>It is according to mixele the control the human divisor applied in<br>the same situation. This lower<br>appear the massive amendment of the same sim or of an expert (a human driver for autonomous driving) by

mismatch, i.e. the algorithm will never encounter failing

cases and thus will not reart appropriately in those condi-<br>tions. Techniques to augment the database with such failing

 $\label{eq:2} \begin{array}{l} \text{abelian} \\ \text{subspace} \\ \text{in } \mathcal{M} \\$ 

**I INTRODUCTION** One of the most challenging problems for autonomous

vehicles is to bandle unsignated intersections in urban envivehicles in lo handle umigrada<br>di interacción, interacción, interacción, la interacción, la interacción, la<br>interacción, interacción, interacción, interacción, interacción, interacción, interacción, interacción, interacci

extend the abilities of autonomous agents and increase safety<br>through driver assistance when a human driver is in control.

through three most<br>states when a human driver is at control. Several stranging income a<br>state how absoluty boson applied to interest the most proposed in the<br>state proposition of the most state of the state is approached

the automotive industry [5]. Variants of the TTC approach

have been used for autonomous driving [6] and the DARPA

This is problematic: in the DARPA Urban Challenge, one

.<br>The University of Pennsylvania, <sup>3</sup>The University of Virginia,<br>Henda Research Institute, The Georgia Institute of Technology

**Ale effectivenes** 

more, we have de

1. Introduction

Maria Huerle<sup>1, c</sup>. Gabriel Kalweit<sup>1, c</sup>. Branka Mirchevska<sup>2</sup>, Moritz Werline<sup>2</sup>, Joschka Boedecker<sup>1,3</sup>



though, special architectural components are pecossary a Using occupancy grids in combination with conneural networks (CNN) imposes a trade-off between comparational workload and copressiveness. Whilst smaller chitectures acting on low-resolution grids as an input or<br>efficient from a commutational recupective, they may be to efficient from a computational perspective, they may be to<br>imprecise to represent the environment correctly. Wherea

.<br>CODE SEE: Feroual use of the matrial is persited. Fermission from SEE must be obtained for all other uses, in any current or future media<br>coluling reprimitionized phi matrial for advertising or presentional proposes, cre

### Model-free Deep Reinforcement Learning for Urban **Autonomous Driving**

Jianya Chen\*, Bodi Yuan\* and Masayoshi Tomizuka

learn how to deal with those

in, which can be coulty and time consuming. 2) It can only<br>the performance of the algorithms, including relationships are also assumed to the algorithms, including relations<br>A One B. Thus and M. Training are with Departmen

### Learning hierarchical behavior and motion planning for **CARLA: An Open Urban Driving Simulator**

EL1

environment perception as well as behavior and motion planner, encoding the traffic situation as a hidden representation.

10 exhaustively characterized

Fermally, we state the driving as histoachical behavior and motion, plated and the state is stated in Fig. 1. The behavior layer makes a decision based on the concern elocation e.g. lane change, while the motion plane in t

night Wass Yes Wan. Deutsch Zester als Bear Norm (2002)<br>18 Mai: San Say Ney Messing of Massai Court an Toden | planting [6]-95] using RL, we set to achieve a policy that<br>19 Mai: San Say Liberatory, Happin, Tal. Chas, Yank ering the coupled action space of motion and behavior, the<br>main challenge is the search inefficiency, especially in the

ensimiente (211). This estima is particularly children in the meaning of the relationship of the meaning in the meaning in the meaning of the mean

to hard. In a<br>protocole at the driving in bindeed by infrastructions costs and the logicities<br>difference of the system of the system of the system of the system of<br>the system of the system of the system of the system of<br>

arvous de la groya as vers la responsa de la groya de sintentistas. Similation can democraties de la groya de la While ad-hoc use of simulation in autonomous driving research is widespread, existing simulation<br>platforms are limited. Open-source racing simulators such as TORCS [28] do not present the com-

1st Conference on Robot Learning (CoRL 2017), Mountain View, United States

any teat Lettourine usupersures enter passang







### **A SAMPLING OF PUBLISHED REWARD FUNCTIONS**

**Paper**

Navigating Occluded Intersections with Autonomous Vehicles using Deep Reinforcement Learning [Isele et al., 2018]

### Unweighted sum of **3 attributes**:

- 0.01 for every step
- 10 if a collision occurred (0 otherwise)
- + 1 when the agent successfully reaches the destination beyond the intersection (0 otherwise)

Deep Distributional Reinforcement Learning Based High-Level Driving Policy Determination [Min et al., 2019]

### Unweighted sum of **4 attributes**:

- $+$  (v-40)/40, where v is speed in km/h within the allowed range [40,80] km/h - 10 if the ego vehicle collides
- (0 otherwise)
- + 0.5 if the ego vehicle overtakes another vehicle (0 otherwise)
- 0.25 if the ego vehicle changes lane (0 otherwise)

CARLA: An open urban driving simulator [Dosovitskiy et al., 2017]

### Weighted sum of **5 attributes**:

- $r = (1) \Delta d + (0.05) \Delta v$ 
	- + (−2\*10−6)Δc + (−2)Δs
	- $+$  (2)  $\Delta$ o
- Δd, the change in distance along the shortest path from start to goal
- $\Delta v$ , the change in speed in km/h
- $\bullet$   $\Delta$ c, the change in collision damage expressed in range [0, 1]
- $\cdot$   $\Delta$ s, the change in the proportion of the ego vehicle overlapping with the sidewalk
- Δo, the change in the proportion of the ego vehicle overlapping with the sidewalk

Learning hierarchical behavior and motion planning for autonomous driving [Wang et al., 2020]

### Defined separately:

- For transitions to terminal states, one of the following:
- + 100 if the goal was reached
- 50 upon a collision or running out of time
- 10 for a red-light violation
- 1 if the ego vehicle is in the wrong lane
- For transitions to non-terminal states, unweighted sum of **3 attr.**:
- $\Sigma_{t}t^{2}[v_{ref}-v(t)]$  /  $\Sigma_{t}t^{2}$ , which rewards speeds close to the desired speed
- 1 /  $[1 + \Sigma_t |v(t)|$ , which rewards based on distance traveled

 $+ \Sigma_{t} [0.02 * d_{\text{olon}}(t)]$  $+ 0.01 * d_{\text{dist}}(t)$ , which rewards keeping larger distances



**Reward Function Reward Function**

# **FIND MISMATCHES IN PREFERENCE ORDERINGS**

(The most powerful method I'm aware of.)

### **7 of 9 reward functions\* incorrectly prefer τcrash.**



W. Bradley Knox, Alessandro Allievi, Holger Banzhaf, Felix Schmitt, and [Peter Stone](http://www.cs.utexas.edu/~pstone). **Reward (Mis)design for Autonomous Driving**. AIJ 2023.

\*9 exhaustively characterized papers' reward functions allow this analysis

Let  $r_{dest}$  be a trajectory that successfully reaches the destination. *τcrash* ≺ *τ idle* ≺ *τdest* .



Let *τ<sub>dest</sub>* be a trajectory that successfully reaches the destination. *τ*<sub>crash</sub> <  $\tau$ <sup>*idle*  $\langle \tau \rangle$ *dest* .</sup>

If  $\tau_{\scriptscriptstyle\mathcal{A}}$  <  $\tau_{\scriptscriptstyle\mathcal{B}}$  <  $\tau_{\scriptscriptstyle\mathcal{C}}$ , then there is some probability  $\rho$  such that  $G(\tau_B) = pG(\tau_C) + (1-p)G(\tau_A)$ 

*p* is the *indifference point*.

For each R, we calculate *p*, then convert it to km per collision at the indifference point.

## Sanity check failure 3: **UNDESIRED RISK TOLERANCE VIA INDIFFERENCE POINTS**











**Indifference points for collision frequency**



**Indifference points for collision frequency**



**Indifference points for collision frequency**





### **OTHER GOTCHAS**

### **● Clipping**

**○ Example: -1,000,000 for collision, +1 for reaching the destination**

### **●** Learnable loopholes

○ when an agent *increases* its utility/return through behavior that *decreases* its performance in the eyes of its designer(s)

**●** Redundant attributes

### Missing attributes  $\rightarrow$  negative side effects



Amodei et al., 2016

# **MINOR SANITY CHECK FAILURES (5–8)**

Identify any of these red flags:

Incomplete description of the problem specification in research presentations

- [Speculation] missing details in a paper indicate not considering reward design to be a critical component of the research project
- 9 of 10 exhaustively characterized papers lacked details of their problem specification (that were learned via our correspondence with their authors)



# **Misalignment by Shaping**

### **REWARD SHAPING**

**Reward shaping**, def. – in addition to the true/environmental reward, providing reward to aid learning, e.g., *by providing behavioral hints or heuristics*

$$
r_{shaped} = R(s_t, a_t, s_{t+1}) + R_{shaping}(s_t, a_t, s_{t+1})
$$

In practice, most RL problems only have one, *shaped* reward function. O

## **REWARD SHAPING**

### What leaders in AI say

—

Russell and Norvig: "As a general rule, it is better to design performance metrics according to what one actually wants to be achieved in the environment, rather than according to how one thinks the agent should behave."

Sutton and Barto agree in almost the same phrasing, adding that imparting knowledge about effective behavior is better done via the initial policy or initial value function.

My version: Specify how to measure *outcomes,* not how to achieve them.

I.e., *in general*, don't shape rewards.

## **REWARD SHAPING**

### What leaders in AI say

—

Russell and Norvig: "As a general rule, it is better to design performance metrics according to what one actually wants to be achieved in the environment, rather than according to how one thinks the agent should behave."

Sutton and Barto agree in almost the same phrasing.

My version: Specify how to measure *outcomes,* not how to achieve them.

I.e., *in general*, don't shape rewards.
## **REWARD SHAPING**

#### "Safe" reward shaping

Safety here means that the reward shaping will not change the optimal policy (or ordering over policies).

Some *specific* methods for reward shaping are safe (under some assumptions), but their desirability is still controversial.

If no effort is made to establish that an instance of reward shaping is safe, then it's unsafe.

## **SAFE REWARD SHAPING METHODS**

"Safe" here means that the reward shaping will not change the optimal policy (or ordering over policies).

- Potential-based reward shaping (Ng. et al, 1999)  $R_{shaping}(s_t, a_t, s_{t+1}) = \gamma \Phi(s_{t+1}) - \Phi(s_t)$ 
	- Its assumptions are often overlooked tabular and a proper task
	- $\circ$  Extension to  $\Phi(s, a)$  (Wiewiora et al., 2023)
	- Equivalence of potential-based shaping and Q-value initialization (Wiewiora et al., 2023)
	- Dynamic potential-based reward shaping (Devlin and Kudenko, 2012)
- Annealing the shaped reward (Behboudian et al., 2020; Szoke et al. 2024)

## REWARD SHAPING IN PRACTICE

### Reward shaping in RL for AD papers

- **●** 13 of 19 include reward shaping
- **●** Some examples of behavior that shaped rewards encourage
	- staying close to the center of the lane [Jaritz et al., 2018]
	- increasing distances from other vehicles [Wang et al., 2020]
	- avoiding overlap with the opposite-direction lane [Dosovitskiy et al., 2017, Liang et al., 2018]

### **REWARD SHAPING IN PRACTICE**

### Reward shaping in RL for AD papers

- **●** Of those 8 exhaustively characterized papers that include reward shaping …
	- 0 explicitly describe the separation of their shaping rewards and their true rewards
	- 0 use a recognized method of safe reward shaping or discuss safety of reward shaping
	- 2 acknowledge usage of reward shaping
	- 1 acknowledges its potential adverse effects

#### Recommendation: **Create an aligned reward function without shaping, then** *optionally* **add a shaping reward function.**

#### **Why?**

- **Clarity** 
	- The reward function should create an aligned problem specification.
	- The shaping rewards give policy guidance and may change the problem specification.
- **● Debugging ("overfit" plot)**



Training episodes

# **Misalignment by Trial-and-Error Design**

# **Imagine you want to design a new RL problem.**

# **Imagine you want to design a new RL problem.**

# **How might you approach this?**

Step 1: Design a candidate RL problem, including R.

Step 1: Design a candidate RL problem, including R.

Step 2: Pick an RL algorithm for testing.

Step 1: Design a candidate RL problem, including R.

Step 2: Pick an RL algorithm for testing.

Step 3: Learn a behavior policy.

Step 1: Design a candidate RL problem, including R.

Step 2: Pick an RL algorithm for testing.

Step 3: Learn a behavior policy.

Step 4: If the policy isn't right, update the RL problem (especially the reward function) and repeat.

## **This trial-and-error process is the norm.**

### **RL for AD**

Of 8 papers whose authors shared their reward design process over email,

**100% used trial-and-error to design** their reward function.

### **General RL experts**

We surveyed 24 expert RL practitioners. **92% used trial-and-error to design** their most recent reward function.



reward design iterations



Trial-and-error reward design iterations

Training context - RL algorithm, hyperparameters, and tasks



Trial-and-error reward design iterations

Training context - RL algorithm, hyperparameters, and tasks



Trial-and-error reward design iterations

# Hungry Thirsty Domain



Singh et al., 2009, Where Do Rewards Come From?

### **Finding the potential for overfitting.**





#### Finding: **Reward functions that achieve the best performance in one learning context can be suboptimal in another.**



For all experiments, we find the best performing reward functions differ across learning contexts.

### This shows potential for overfitting.

### **H2: The cumulative performances achieved with different reward functions are** *uncorrelated* **across different learning contexts.**



We rank all reward functions for each experiment setting  $(D_{\mathbb{R}} \quad D_2)$ 

We compare the ordering of these rankings using Kendall's tau.

Distribution Sample

#### Finding: **The cumulative performances achieved with different reward functions are uncorrelated across different learning contexts.**



We rank all reward functions for each experiment setting  $(D_{\mathbf{X}} \cdot \mathbf{D}_2)$ 

We compare the ordering of these rankings using Kendall's tau.

We find that these rankings are **uncorrelated** ( $|\tau_b|$  < 0.1) or **slightly correlated (** $|\tau_b|$  **< 0.2).** 

# **User Study Conducted in Jupyter Notebooks**





#### **Experts overfit reward functions too**

# **68%** of users overfit reward functions

User P20 first tried a reward function which achieved a mean score of 138,092 with DDQN.

They ultimately selected a different reward function, which achieved a mean score of 1,031 with DDQN.

# Experts' reward functions tend to not generalize.



**Hard** configuration

(15 steps between water & food)



**Easy** configuration

(5 steps between water & food)

# Experts' reward functions tend to not generalize.



53% of RL experts submitted reward functions that had optimal policies which do not perform the hard configuration well.

**Hard** configuration

(15 steps between water & food)

# Experts are currently bad at writing reward functions.



53% of RL experts wrote reward functions which **failed to encode the task** in the hard case.

For example, **P3**'s reward function:

$$
\begin{array}{cc} r(\neg{\tt H}\wedge\neg{\tt T})=1.0 & r({\tt H}\wedge\neg{\tt T})=-0.1 \\ r(\neg{\tt H}\wedge{\tt T})=1.0 & r({\tt H}\wedge{\tt T})=-1.0 \end{array}
$$

**Hard** configuration

(15 steps between water & food)

Most experts (83%) used a *myopic* design strategy of using reward to order states by their desirability.

#### **Example**

"It's best to not be hungry and thirsty, so I'll set that to the max, 1. Being not thirsty is better than being not hungry [so 0.3 for only hungry /not thirsty and -0.35 for only thirsty / not hungry]. Worst is at hungry AND thirsty; setting that to  $-1''$  – Participant 25

*Is the word "reward" harming reward design?*

# Reasoning about reward accumulation (return) is done poorly.

#### **Example reused**

"It's best to not be hungry and thirsty, so I'll set that to the max, 1. Being not thirsty is better than being not hungry [so 0.3 for only hungry /not thirsty and -0.35 for only thirsty / not hungry]. Worst is at hungry AND thirsty; setting that to  $-1''$  – Participant 25



Trial-and-error reward design can overfit to the training context (RL algorithm, hyperparameters, and task).

And RL experts appear to do so in practice. *Impact on algorithmic comparison and ablation studies?*

# **Misalignment by Discounting**

## **REWARD AND RETURN**

#### **Field**

-

-

reinf. learning motion planning control theory evolutionary algs. utility theory optimization



\* "objective function" more precisely refers to the expectation of  $G(\tau)$ 

### **REWARD AND RETURN**

 $T$  $G(\tau) = \sum \gamma^t R(s_t, a_t, s_{t+1})$  $t=0$ 

### **REWARD AND RETURN**

 $T$  $G(\tau) = \sum \gamma^t R(s_t, a_t, s_{t+1})$  $t=0$ **discount factor**



**Time steps until a reward**

### **Contemporary RL tends to have 2 discount factors: problem-side and algorithmic**

#### **Problem-side, γ<sub>MDP</sub>** - part of the MDP definition

- determines how return should be calculated when evaluating a policy's performance (e.g., for comparing algorithms or reporting results in a publication)
- with a start state distribution, determines the ranking of policies and therefore the set of optimal policies

#### **Algorithmic, γ<sub>alg</sub>** - a hyperparameter of the RL algorithm

- $\bullet$  *Y***alg**  $\leq$  *Y***MDP**
- **γalg** ≤ 0.999 in deep RL papers I have seen, usually **γalg** ≤ 0.99
- in practice, **γalg** trades stability during learning at the cost of greater distance between the RL algorithm's loss function and the task objective

#### **Do not confuse the two! We focus on γ<sub>MDP</sub> unless otherwise stated.**
### **Contemporary RL tends to have 2 discount factors: problem-side and algorithmic**

**Problem-side,**  $Y_{MDP}$  - part of the RL problem definition

**Exercise creates the true return** 

**Algorithmic, γ<sub>alg</sub>** - a hyperparameter of the RL algorithm

- $\bullet$   $\mathsf{Y}_{\text{alg}} \leq \mathsf{Y}_{\text{MDP}}$
- **γalg** ≤ 0.999 in deep RL papers I have seen, usually **γalg** ≤ 0.99
- in practice, **γalg** trades stability during learning at the cost of greater distance between the RL algorithm's loss function and the task objective

### **Do not confuse the two! We focus on γ<sub>MDP</sub> unless otherwise stated.**

### **Estimating return during RL at absorbing state vs. when stopping an episode for other reasons**

**If stopping at absorbing state**—i.e., satisfying termination conditions—the absorbing state value is 0 except under highly unusual circumstances.

When function approximation is used, there is danger that value inference will return a nonzero value. You can use  $y=0$  to get the equivalent effect as having a value of 0.

**If stopping at non-absorbing state**—i.e., without satisfying termination conditions—include the value of the final state discounted by  $Y_{\text{alg}}$  (or  $Y_{\text{MDP}}$ ).



### **The set of optimal policies can change as the discount factor changes.**



In this continuing domain,

- $\bullet$  if  $y < 0.5$ , then choosing the left loop from s is optimal
- $\bullet$  if  $\gamma > 0.5$ , then choosing the fight loop from s is optimal

**Separate intuitive argument:** if changing γ didn't change the set of optimal policies, then we would just set  $y=0$  and forget about the credit assignment problem.

## **To develop intuition about your discounting, calculate time-to-10% value (and 1% and 0.1%) via log<sup>γ</sup>**

**Example:** Autonomous driving often has 100ms time steps.

#### If **γ=0.9**,

the rewards are discounted to X% of their full value this far in the future:

- 10% 2.19 s = *log<sup>γ</sup> 0.1 \* 0.1s*
- 1% 4.37 s = *log<sup>γ</sup> 0.01 \* 0.1s*
- 0.1% 6.56 s = *log<sup>γ</sup> 0.001 \* 0.1s*

It takes a constant amount of time for each reduction by a factor 0.1.

## **To develop intuition about your discounting, calculate time-to-10% value (and 1% and 0.1%) via log<sup>γ</sup>**

**Example:** Autonomous driving often has 100ms time steps.

#### If **γ=0.99**,

the rewards are discounted to X% of their full value this far in the future:

- $\bullet$  **10% 22.9s** = *log<sub>γ</sub>*0.1 \* 0.1s
- 1% 45.8s = *log<sup>γ</sup> 0.01 \* 0.1s*
- 0.1% 68.7s = *log<sup>γ</sup> 0.001 \* 0.1s*

Even with a relatively high γ=0.99, events one minute into the future likely have negligible effect on the value function!

## **To develop intuition about your discounting, calculate time-to-10% value (and 1% and 0.1%) via log<sup>γ</sup>**

**Example:** Autonomous driving often has 100ms time steps.

#### If **γ=0.999**,

the rewards are discounted to X% of their full value this far in the future:

- 10% 230 s / 4 min
- 1% 460 s / 8 min
- 0.1% 690 s / 12 min

Each 10x decrease in  $(1 - y)$  results in a  $\sim$ 10x increase in horizon.

While a precise horizon does not exist, there is an order of magnitude in which discounting goes from being significant to being negligible.

### **Make all episodic tasks undiscounted.**

**Exponential discounting** is a seemingly necessary evil in continuing tasks. It ensures finite returns and encourages getting reward sooner.

#### But it has **drawbacks**:

 1) It appears to **decrease alignment** with humans, who *do not evaluate outcomes with exponential discounting.*

2) It **makes return less legible** / human-readable.

It's not necessary though in episodic tasks, so to avoid these drawbacks it should not be used.

Whether you use discounting for your *algorithm* is a different matter.

### **A continuing exponentially discounted task may not have an optimal policy under function approximation.**



Argument comes from **Discounted Reinforcement Learning Is Not an Optimization Problem** by Naik et al. (2019).

### **A continuing exponentially discounted task may not have an optimal policy under function approximation.**



Argument comes from **Discounted Reinforcement Learning Is Not an Optimization Problem** by Naik et al. (2019).

### **A continuing exponentially discounted task may not have an optimal policy under function approximation.**



Under the optimality criterion **vπ (s) ≥ vπ'(s)** *for all states s and all policies π',* there may be no optimal *representable* policy.

Can we still specify an optimal *representable* policy by setting the start state distribution? I.e., set  $J(\pi) = \mathbb{E}_{s_0 \sim p(s_0)}[v_\pi(s_0)]$ 

#### **Not if we want an aligned learning objective.**

- Over the infinite time of a continuing task, the state visitation distribution may have no support for the states that are visited from start states within the discount factor's "horizon" of non-negligible impact.
- Generally violates the idea that we care about performance over an infinite task, not just at its start.

# **Designing Aligned Reward**

**There are no best practices!**  (Well, not yet.)

### **But our methods for catching misalignment might help.**

### **Sketch of possible best practices**

- 1. Consider the simplest set of **outcome variables** that differentiate varying levels of success vs. failure.
	- Find a per-time step version of each outcome variable that adds up to its full-trajectory value.
	- Example: time to goal
	- Example: soccer
- 2. Create a **parametrized reward function representation** with these variables.
	- Recommendation: try a linear representation and stubbornly try to make it work
- 3. **Tune the parameters** so that its preference ordering over outcome distributions matches yours.
- 4. **Evaluate**.

*At any point, you may learn something that causes you to return to an earlier step.*

### **Sketch of possible best practices**

- 1. Consider the simplest set of outcome variables that differentiate varying levels of success vs. failure.
	- Find a per-time step version of each outcome variable that adds up to its full-trajectory value.
	- Example: time to goal
	- Example: soccer
- 2. Create a parametrized reward function representation with these variables.
	- Recommendation: try a linear representation and stubbornly try to make it work
- **3. Tune the parameters so that its preference ordering over outcome distributions matches yours.**
- 4. Evaluate.

*At any point, you may learn something that causes you to return to an earlier step.*

### **Sketch of possible best practices**

1. Consider the simplest set of outcome variables that differentiate varying levels of success vs. failure.

○ Find a per-time step version of each outcome variable that adds up to its full-trajectory **Methods for finding misalignment become methods for optimizing** 

- o Example: time to the reward function via RLHF.
- 
- 2. Create a parametrized reward function representation with these variables.
	- Recommendation: try a linear representation and stubbornly try to make it work
- **3. Tune the parameters so that its preference ordering over outcome distributions matches yours.**
- 4. Evaluate.

At any point, you may learn something that causes you to return to an earlier step.



### **Tools and insights**

- Catching misalignment via preference mismatch (R vs. human)
	- Preferences over trajectorie
	- Preferences over trajectory lotteries
- Shaping
	- Keep shaping in a separate function.
	- Plot true return vs. shaped return to detect overfitting.
	- Consider a shaping method that doesn't change the preference ordering over policies / outcome distributions.
- **Discounting** 
	- Keep a separate problem-side γ.
	- Calculate time to 10% / 1% / 0.1% value.
- **Others** 
	- Consider how the RL alg modifies R e.g., clipping
	- Bias towards designing an R with legible return and expected return

## **What Should We Work Towards?**

### **Promising projects**

- Validated best practices for aligned reward function design
- When reward cannot be practically aligned
	- Some valued outcomes aren't measurable by the learning system
- Debug methods --> debug tools

**AI safety agenda**: Expose where reward design can cause dangerous misalignment. Fix it if possible. Otherwise, identify where it should not be used.

### **Reward design consultation**

Free for academic or non-profit projects

### Designing a reward function?

Want to talk about it with someone?



#### Email bradknox@cs.utexas.edu and ask for a 30 minute consultation.

Disclaimer: I research the design of reward functions. I want to help you while developing my methodology for doing these consultations, which may eventually be published as best practices for reward function design. This is not a formal investigation, but I do hope to learn from you what was helpful and what was not.

### **Our papers on reward design**

W. Bradley Knox, Alessandro Allievi, Holger Banzhaf, Felix Schmitt, and [Peter Stone](http://www.cs.utexas.edu/~pstone). **Reward (Mis)design for Autonomous Driving**. AII 2023.

Serena Booth, W. Bradley Knox, Julie Shah, Scott Niekum, Peter Stone, Alessandro Allievi. **The perils of trial-and-error reward design: misdesign through overfitting and invalid task specifications.** AAAI 2023.

*Brad Knox <bradknox@cs.utexas.edu>*