How to Specify *Aligned* Reinforcement Learning Problems

Brad Knox

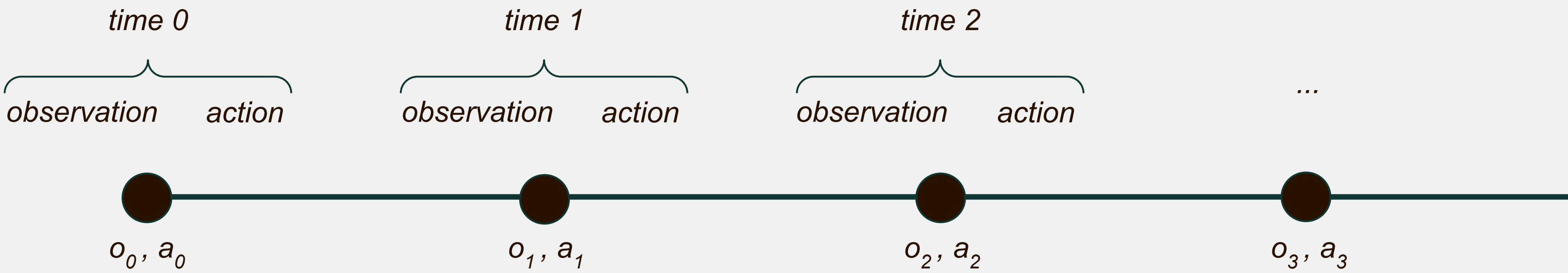
TEXAS
The University of Texas at Austin

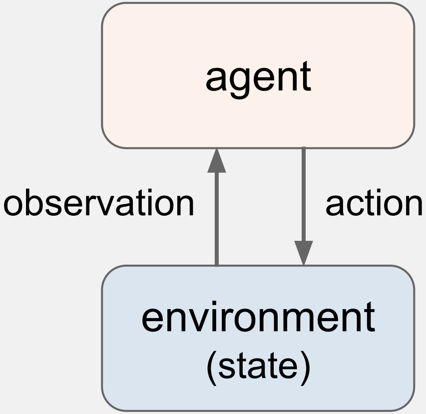
Outline

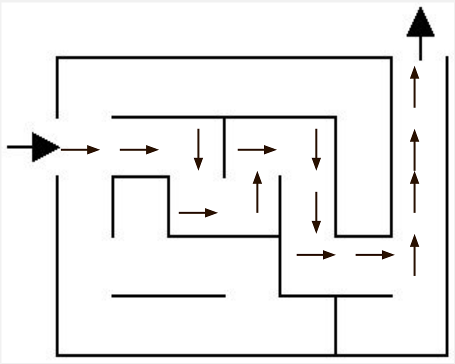
1. Aligned reward functions
2. Finding misalignment in a reward function
3. Misalignment by shaping
4. Misalignment by trial-and-error design
5. Misalignment by discounting
6. Designing aligned reward functions
7. What should we work towards?

Aligned reward functions

Sequential Decision-Making

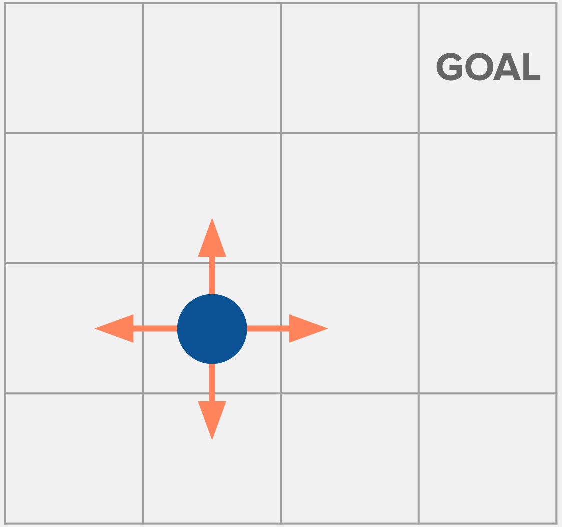






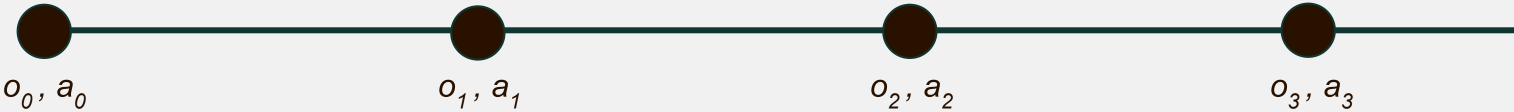
Sequential Decision-Making

Gridworld



States: cell location

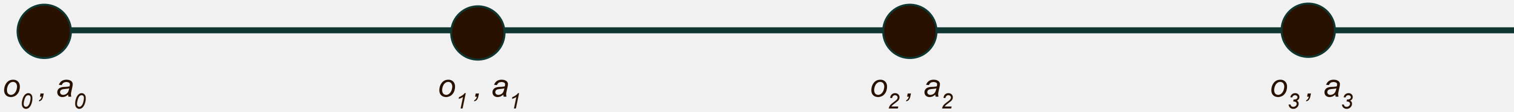
Actions: move in each cardinal direction



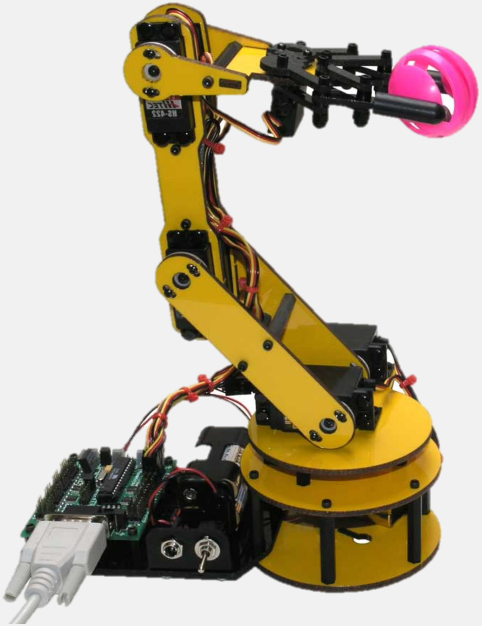
Sequential Decision-Making

State: board configuration

Action: a legal move



Sequential Decision-Making

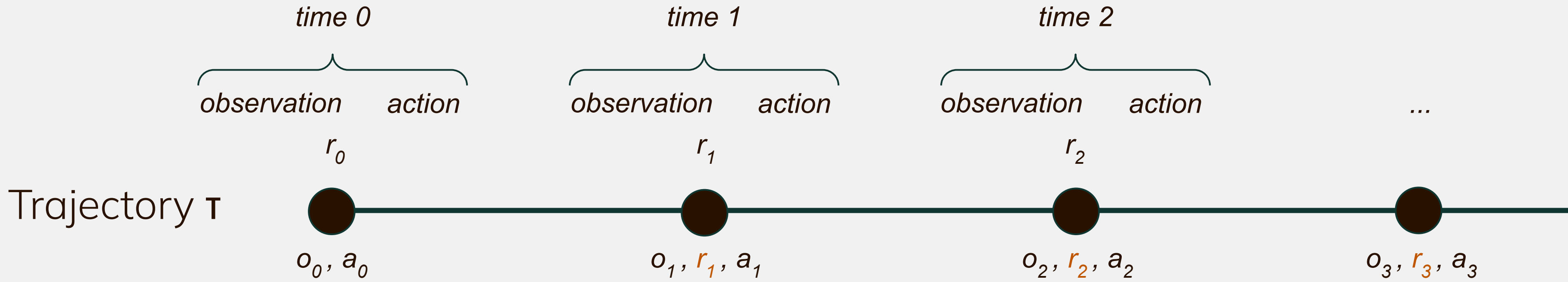


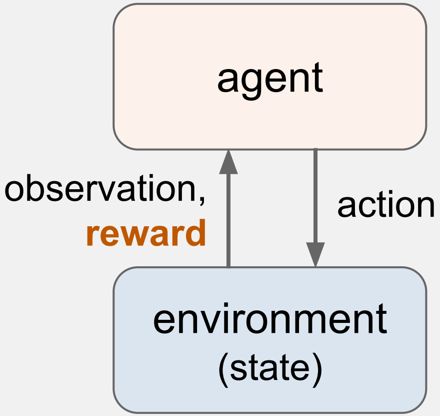
Observation: joint positions, joint velocities, and a camera image of its task area

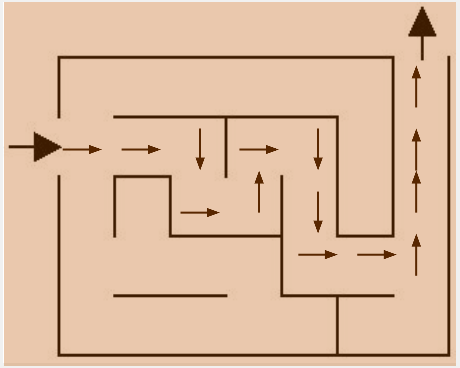
Action: acceleration on each joint

o_0, a_0 o_1, a_1 o_2, a_2 o_3, a_3

Reinforcement learning



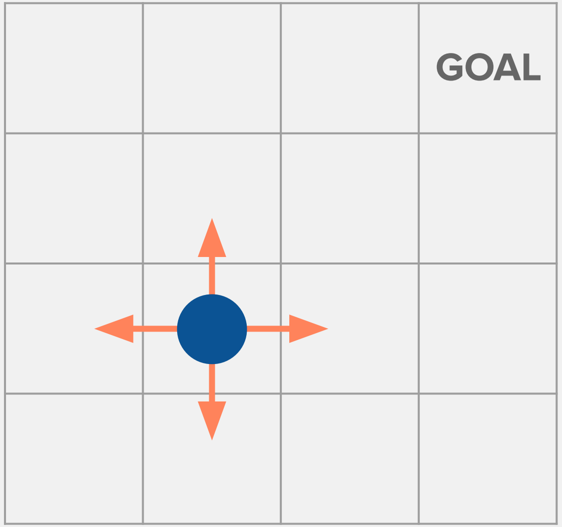




An RL algorithm tries to learn behavior that increases its long-term accumulation of reward.

Reinforcement learning

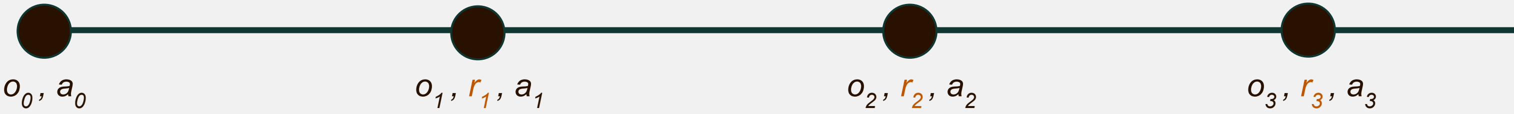
Gridworld



State: cell location

Action: move in each cardinal direction

Reward: -1 per time step / action

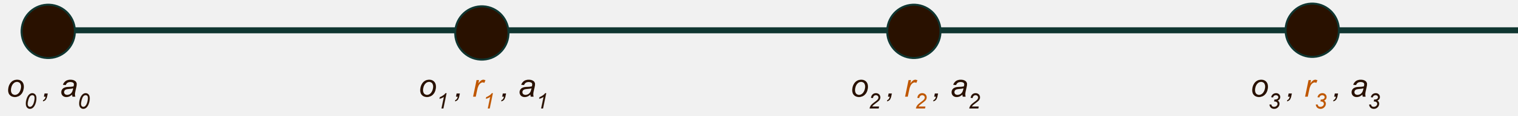


Reinforcement learning

State: board configuration

Action: a legal move

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



REWARD AND RETURN

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

REWARD AND RETURN

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

Field		
reinf. learning	return	reward
motion planning	-1 × cost	-1 × cost
control theory	-1 × cost	-1 × cost
evolutionary algs.	fitness	-
utility theory	utility	-
optimization	objective function*	-
-	performance metric	-
-	score	-

* "objective function" more precisely refers to the expectation of G(τ)

REWARD AND RETURN

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

discount factor

EXPECTED RETURN

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

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

REWARD AND RETURN

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

discount factor

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

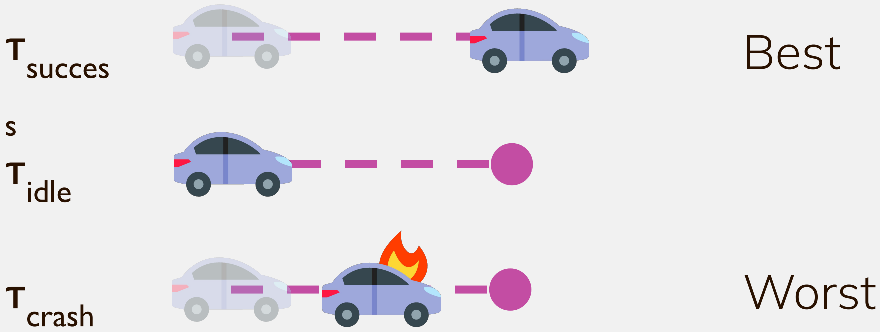
A **reward function** creates a preference ordering over possible trajectories (by $G(\tau)$) and probability distributions over trajectories.

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.

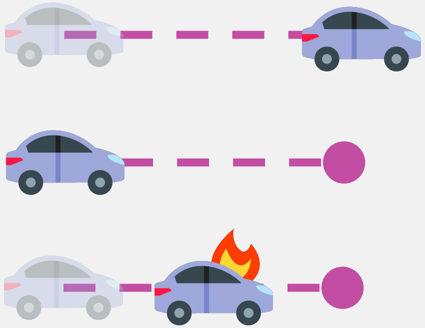
An aligned reward function



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

An aligned reward function

<u>Human stakeholder</u>		<u>Return</u>
Best	$\tau_{success}$	10
↕	τ_{idle}	0
Worst	τ_{crash}	-50



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

An aligned reward function?

<u>Human stakeholder</u>

<u>Expected</u> <u>return</u>

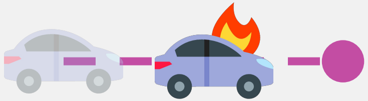
Best

$\tau_{success}$  10

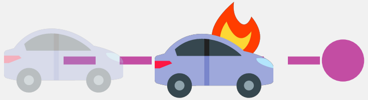
τ_{idle}  0

Lottery: 90% $\tau_{success}$, 10% τ_{crash}

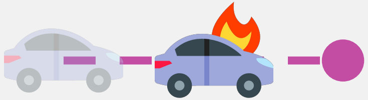
Worst

τ_{crash}  -50

An aligned reward function?

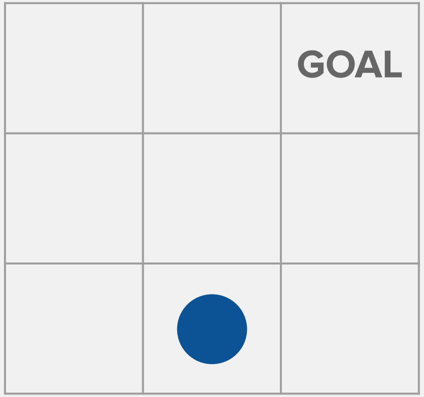
Human stakeholder		Expected return
Best	$\tau_{success}$ 	10
	τ_{idle} 	0
	Lottery: 90% $\tau_{success}$, 10% τ_{crash}	4
Worst	τ_{crash} 	-50

An aligned reward function?

Human stakeholder			Expected return	
Best	$\tau_{success}$		10	
	τ_{idle}		0	Mismatched with human ordering!
	Lottery: 90% $\tau_{success}$, 10% τ_{crash}		4	
Worst	τ_{crash}		-50	

Design a reward function

Gridworld



State: cell location

Action: move in a cardinal direction

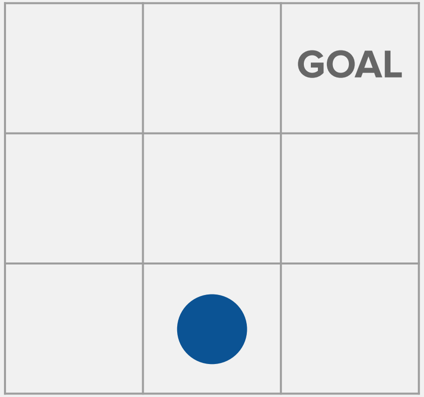
The agent's purpose is to reach the goal in the minimum mean time from any state.

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).

Design a reward function

Gridworld



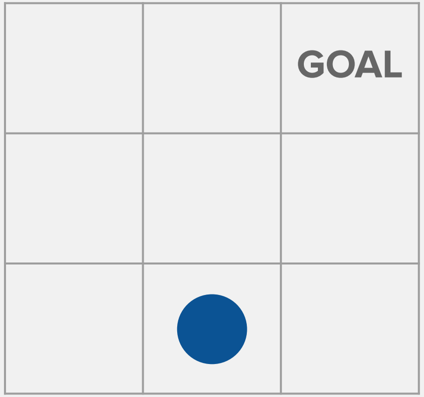
An aligned reward function

-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.

Design a reward function

Gridworld



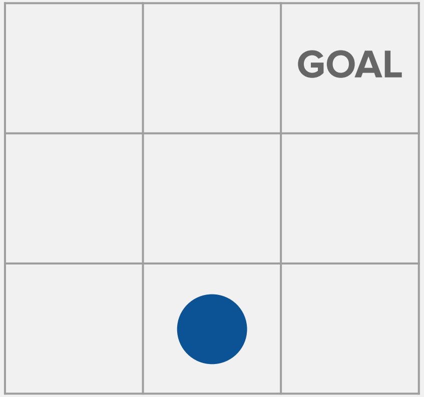
A misaligned reward function

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.

Design a reward function

Gridworld



A misaligned reward function

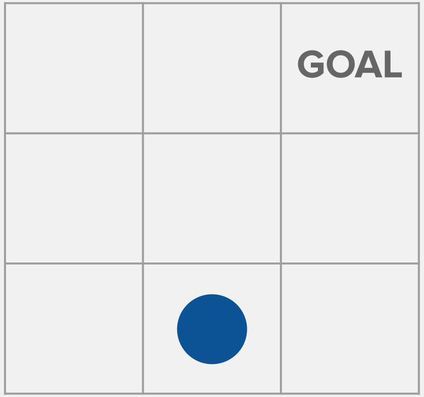
1 reward for an action towards the goal
0 reward otherwise

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?

Design a reward function

Gridworld



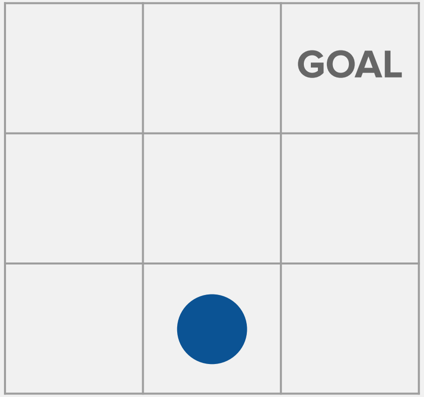
A misaligned reward function

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.

Design a reward function

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?

Science
JOURNALS AAAS

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[1,2], Noam Kolt[3,4], Yoshua Bengio[5,6], Gillian K. Hadfield[2,3,4,7], Stuart Russell[1,2]

Technical 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 regu-

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

"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?

Science
JOURNALS AAAS

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**[1,2], **Noam Kolt**[3,4], **Yoshua Bengio**[5,6], **Gillian K. Hadfield**[2,3,4,7], **Stuart Russell**[1,2]

Technical 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 regu-

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

"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?

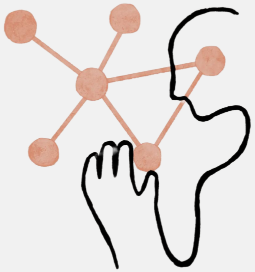
Observation: prompt + any previous text

Action: n-token response

Reward: ???

o_0, a_0 — o_1, r_1, a_1 — o_2, r_2, a_2 — o_3, r_3, a_3

Is RL impactful?

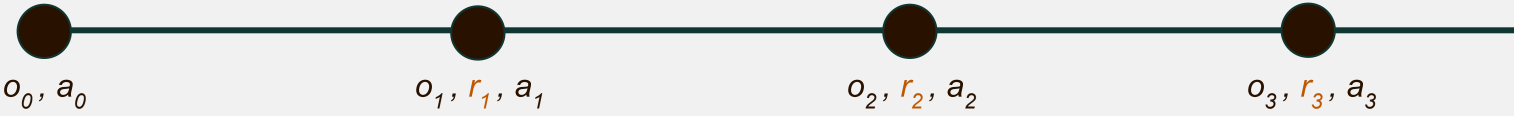


Observation: prompt + any previous text

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.

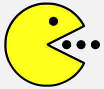
Does not include typical episodic tasks, such as those with goal or failure states.

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:

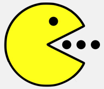
- Agent is a pursuer of reward. 
- Agent estimates expected return from the reward it has experienced.
- Agent identifies actions (more precisely, changes in policy) that will increase estimated expected return.

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. 
- 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

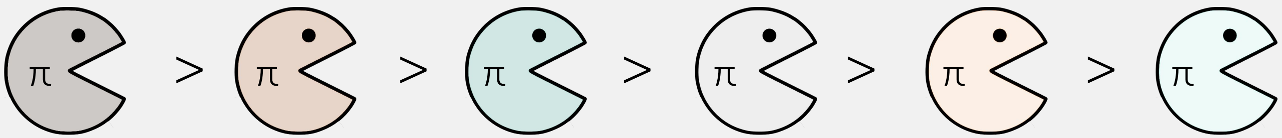
A reward-centric perspective of policies

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.

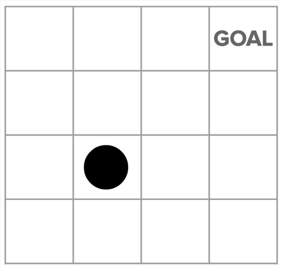
Two highly common reward functions have interpretable return.

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.

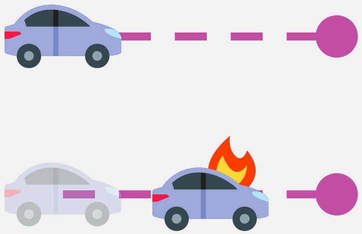
So the task is to maximize the probability of winning.

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 τ_A is preferable to τ_B (i.e., $\tau_A \succ \tau_B$), then return of τ_A > return of τ_B should hold.

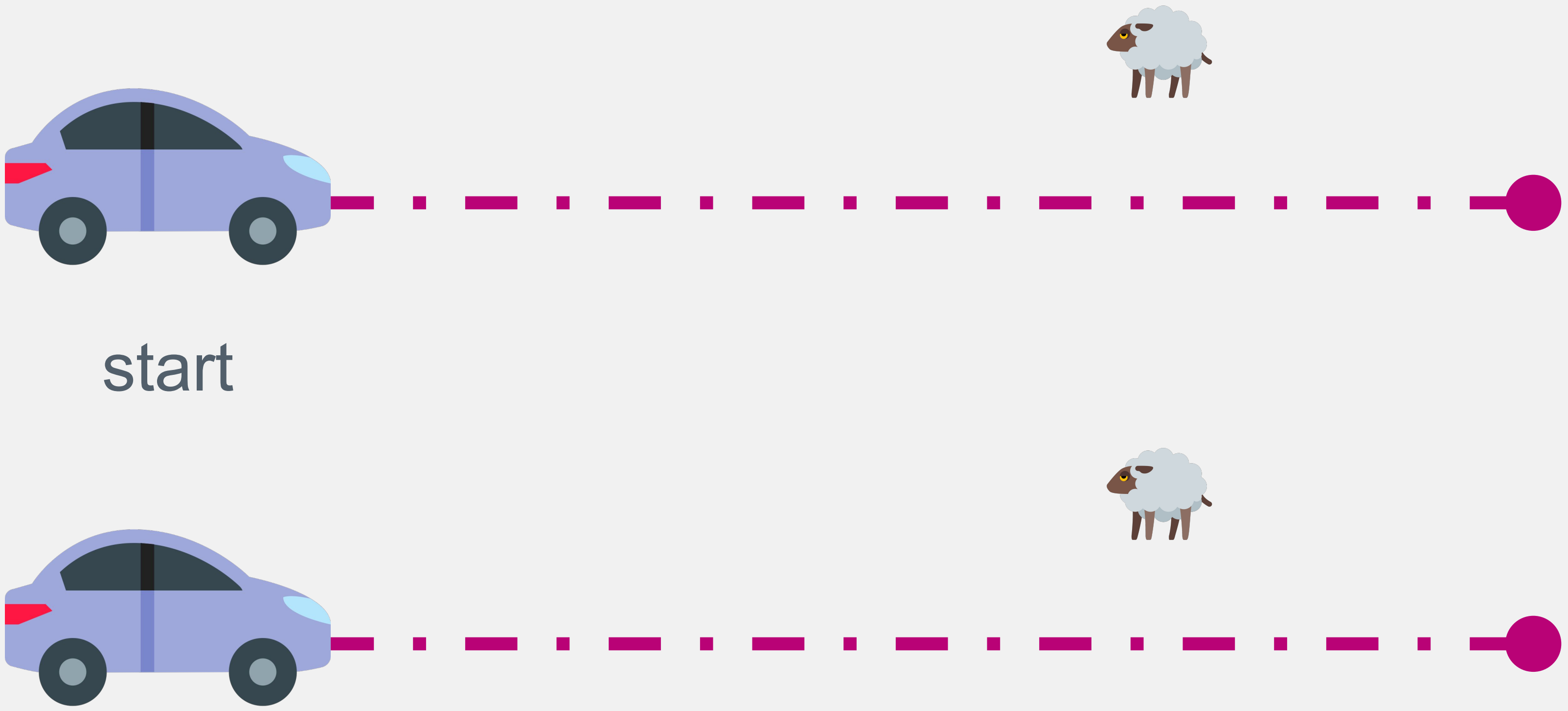


τ_{idle}

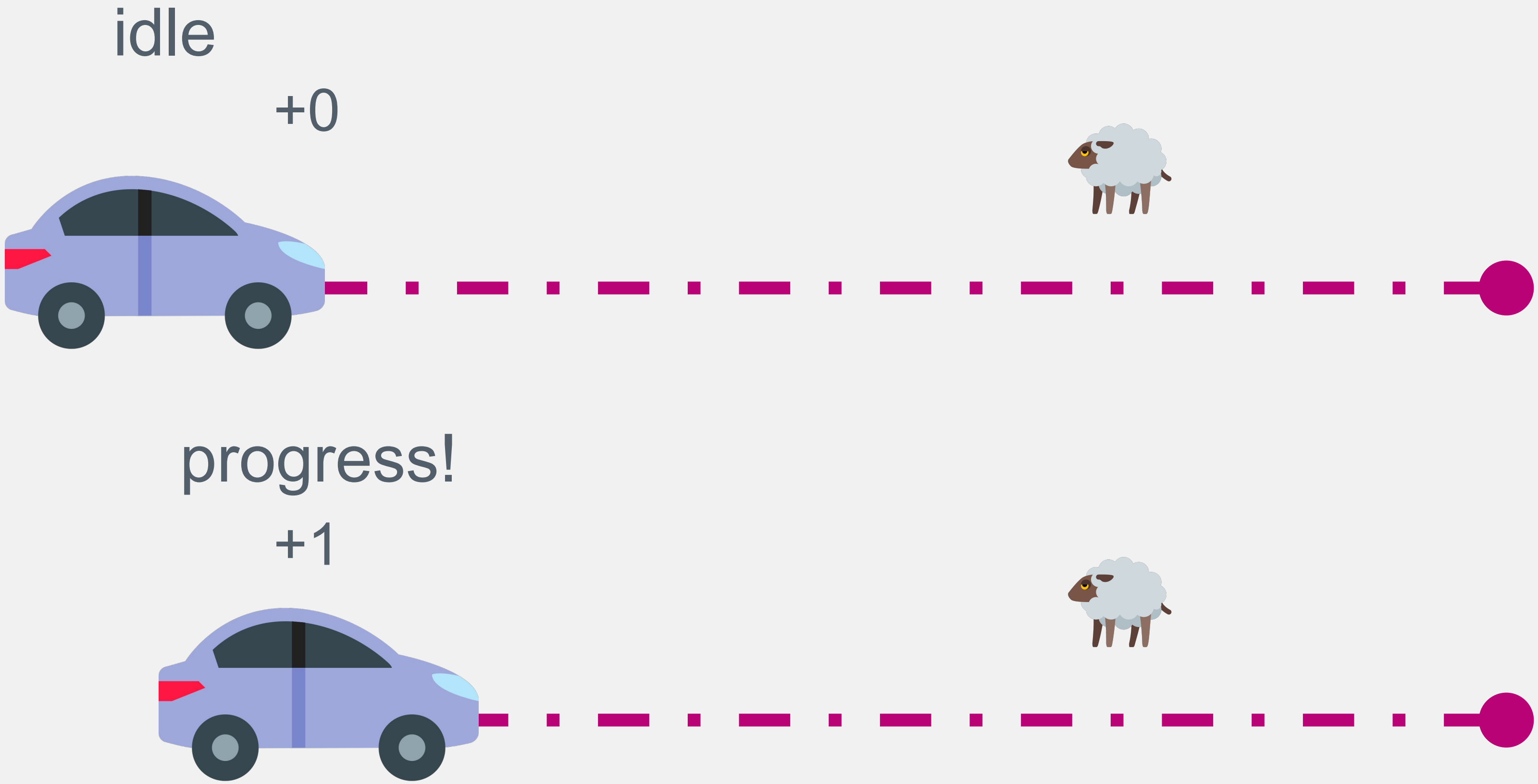
τ_{crash}

W. Bradley Knox, Alessandro Allievi, Holger Banzhaf, Felix Schmitt, and Peter Stone. **Reward (Mis)design for Autonomous Driving**. AIJ 2023.

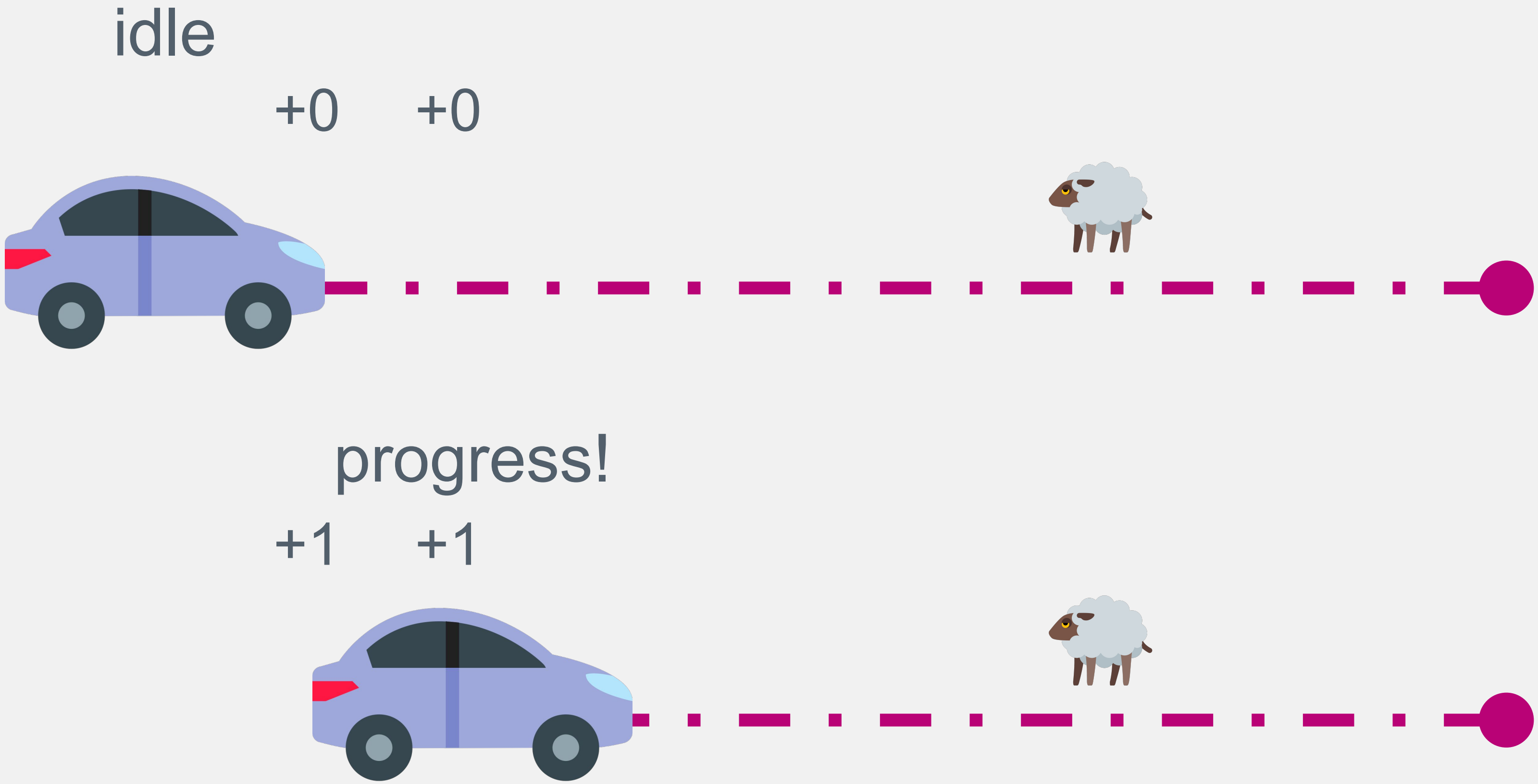
Rewarding two drives



Rewarding two drives



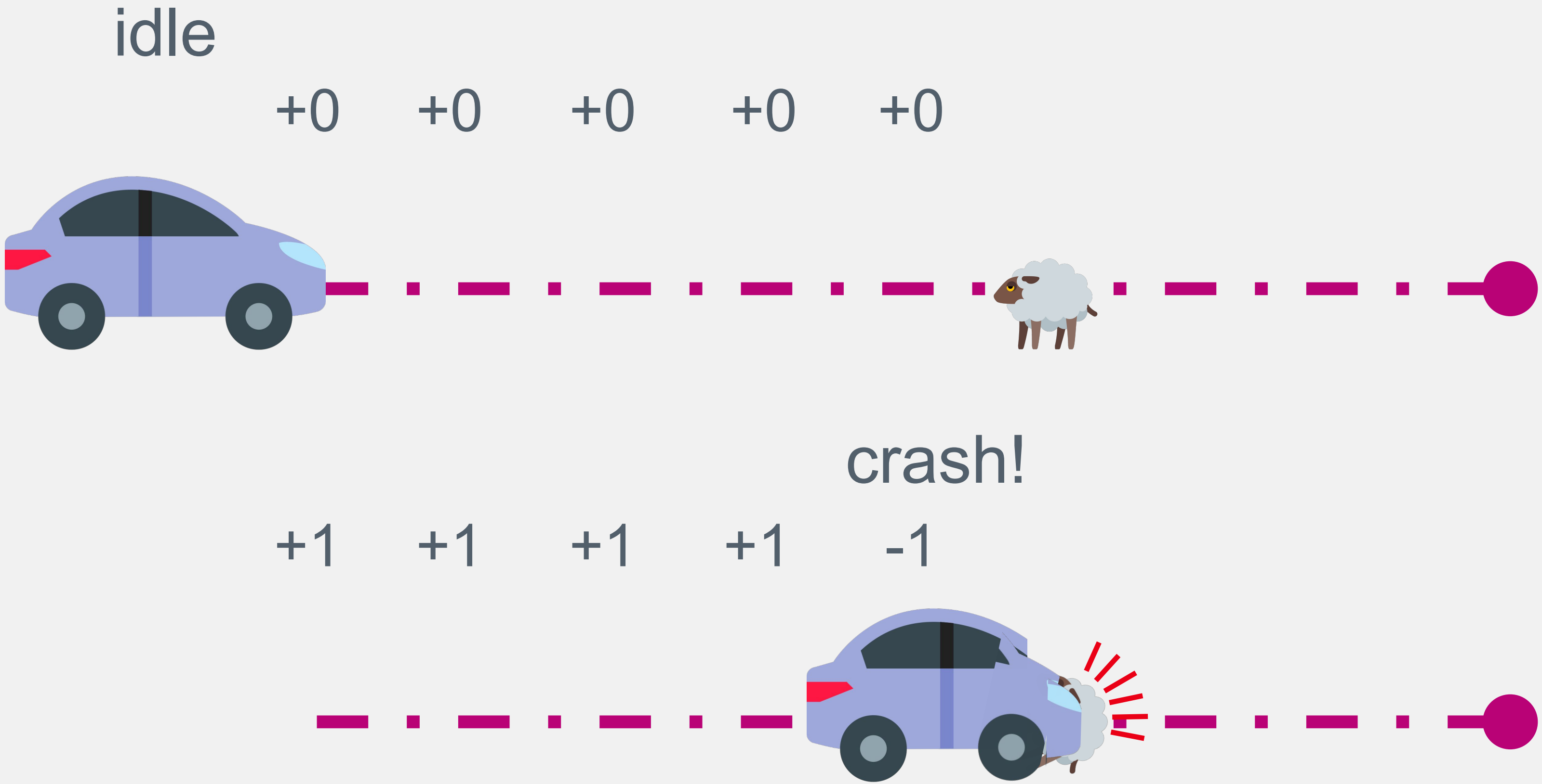
Rewarding two drives



Rewarding two drives

Rewarding two drives

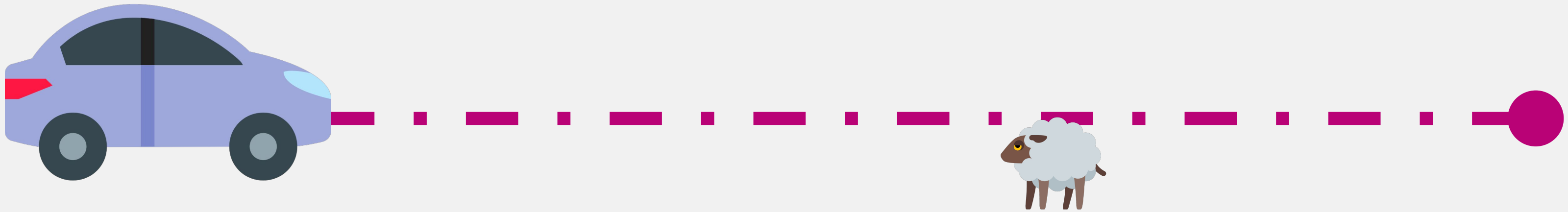
Rewarding two drives



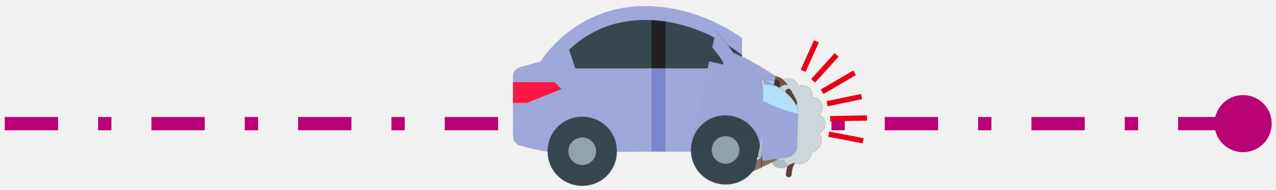
Rewarding two drives

idle

+0 +0 +0 +0 +0 = 0



+1 +1 +1 +1 -1 = +3



Rewarding two drives

idle

+0 +0 +0 +0 +0 = 0

The reward function prefers this drive!!

+1 +1 +1 +1 -1 = +3

Published reward functions for autonomous driving

19 papers reviewed

10 exhaustively characterized

TEXAS
The University of Texas at Austin

BOSCH

A SAMPLING OF PUBLISHED REWARD FUNCTIONS

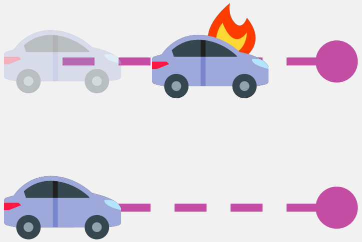
Paper	Navigating Occluded Intersections with Autonomous Vehicles using Deep Reinforcement Learning [Isele et al., 2018]	Deep Distributional Reinforcement Learning Based High-Level Driving Policy Determination [Min et al., 2019]	CARLA: An open urban driving simulator [Dosovitskiy et al., 2017]	Learning hierarchical behavior and motion planning for autonomous driving [Wang et al., 2020]
Reward Function	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)	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)	Weighted sum of **5 attributes**: $r = (1)\Delta d + (0.05)\Delta v + (-2*10^{-6})\Delta c + (-2)\Delta s + (2)\Delta o$ • Δd, the change in distance along the shortest path from start to goal • Δv, the change in speed in km/h • Δc, the change in collision damage expressed in range $[0,1]$ • Δs, 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	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 - $1 \ / \ [1 + \Sigma_t \lvert v(t) \rvert$, which rewards based on distance traveled + $\Sigma_t[0.02*d_{olon}(t) + 0.01*d_{olat}(t)]$, which rewards keeping larger distances

TEXAS The University of Texas at Austin BOSCH

FIND MISMATCHES IN PREFERENCE ORDERINGS

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

7 of 9 reward functions* incorrectly prefer τ_{crash}.



τ_{crash}

τ_{idle}

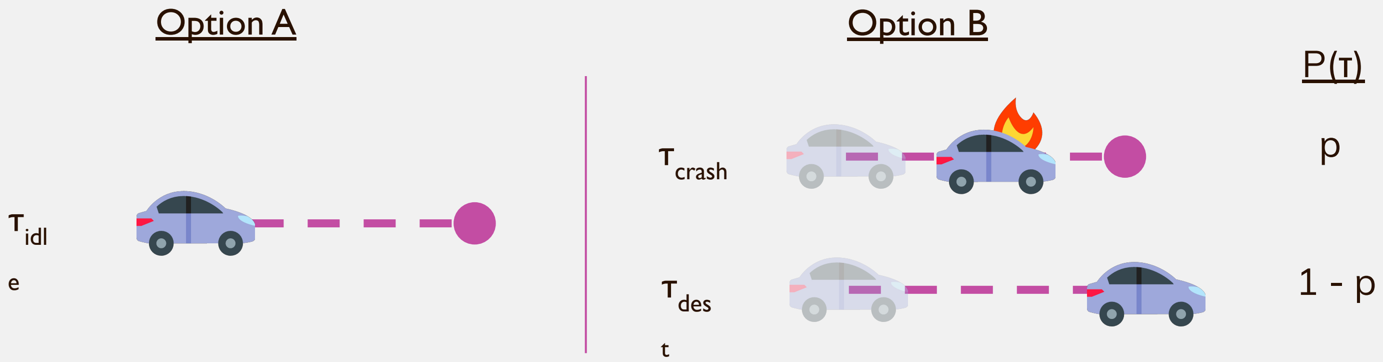
W. Bradley Knox, Alessandro Allievi, Holger Banzhaf, Felix Schmitt, and Peter Stone. **Reward (Mis)design for Autonomous Driving**. AIJ 2023.

*9 exhaustively characterized papers' reward functions allow this analysis

FIND UNDESIRED RISK TOLERANCE VIA INDIFFERENCE POINTS

Let τ_{dest} be a trajectory that successfully reaches the destination.

$\tau_{crash} \prec \tau_{idle} \prec \tau_{dest}$.



FIND UNDESIRED RISK TOLERANCE VIA INDIFFERENCE POINTS

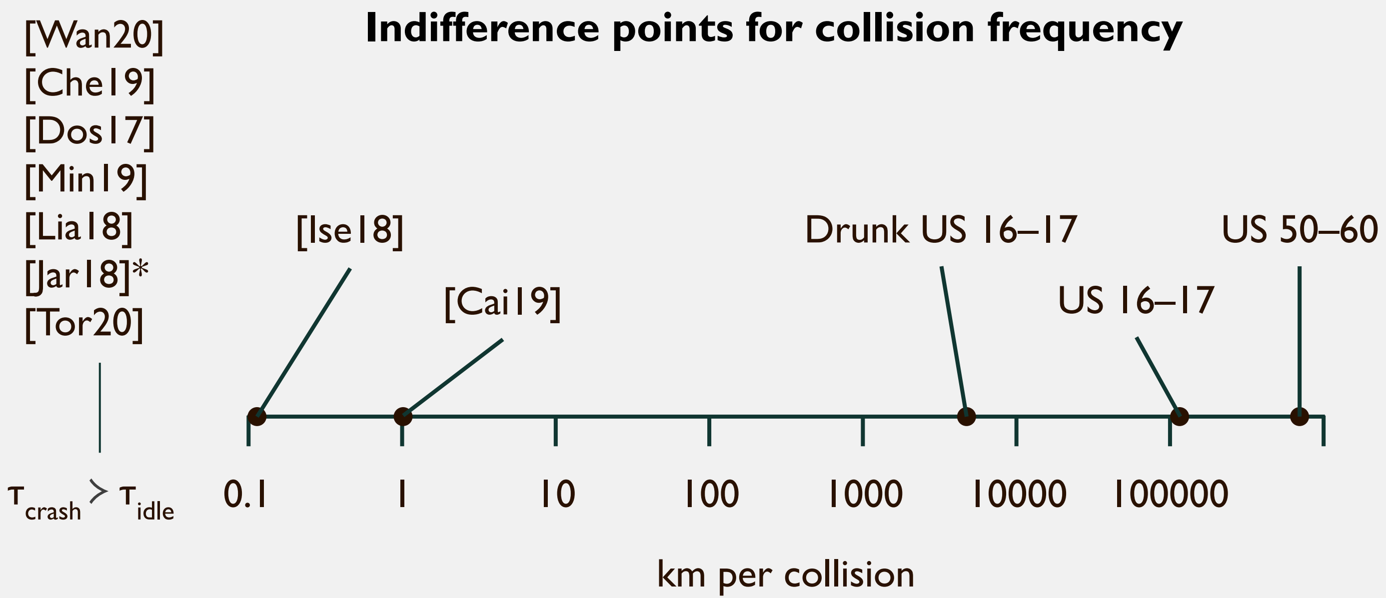
Let τ_{dest} be a trajectory that successfully reaches the destination. $\tau_{crash} \prec \tau_{idle} \prec \tau_{dest}$.

If $\tau_A \prec \tau_B \prec \tau_C$, then there is some probability p such that
$$G(\tau_B) = pG(\tau_C) + (1-p)G(\tau_A)$$

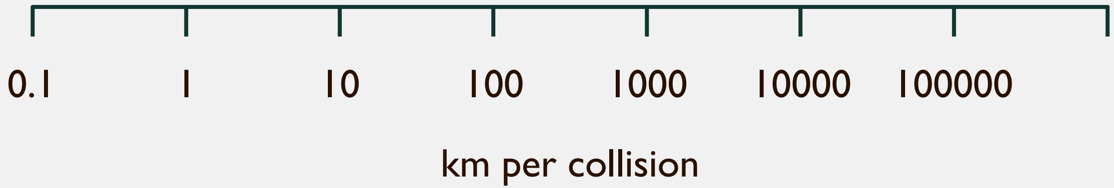
p is the *indifference point*.

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

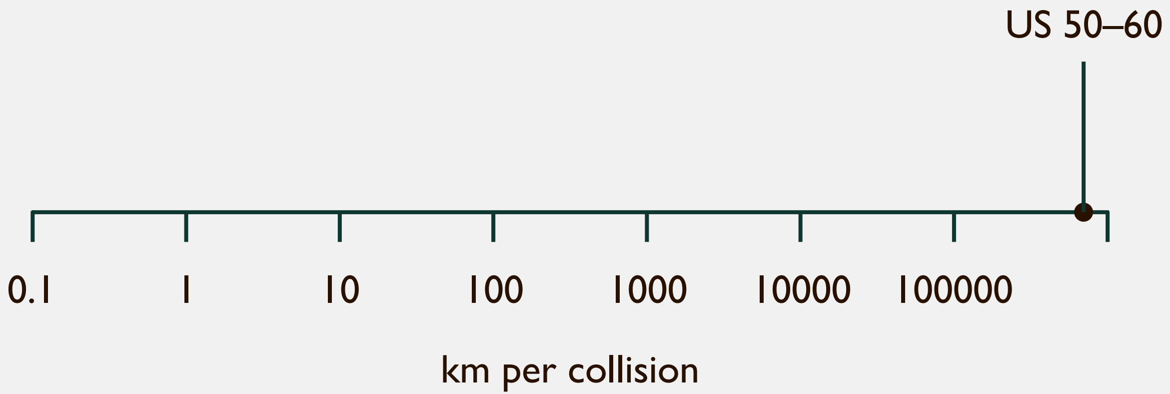
Sanity check failure 3: **UNDESIRED RISK TOLERANCE VIA INDIFFERENCE POINTS**



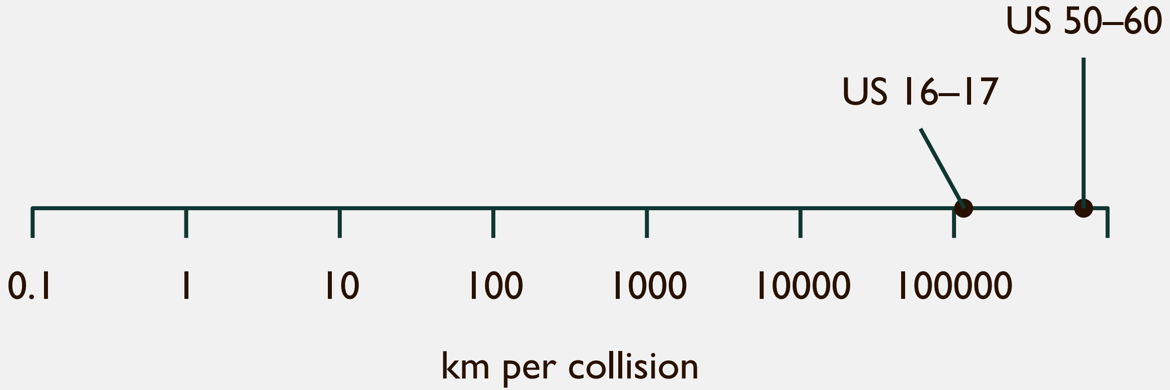
FIND UNDESIRED RISK TOLERANCE VIA INDIFFERENCE POINTS



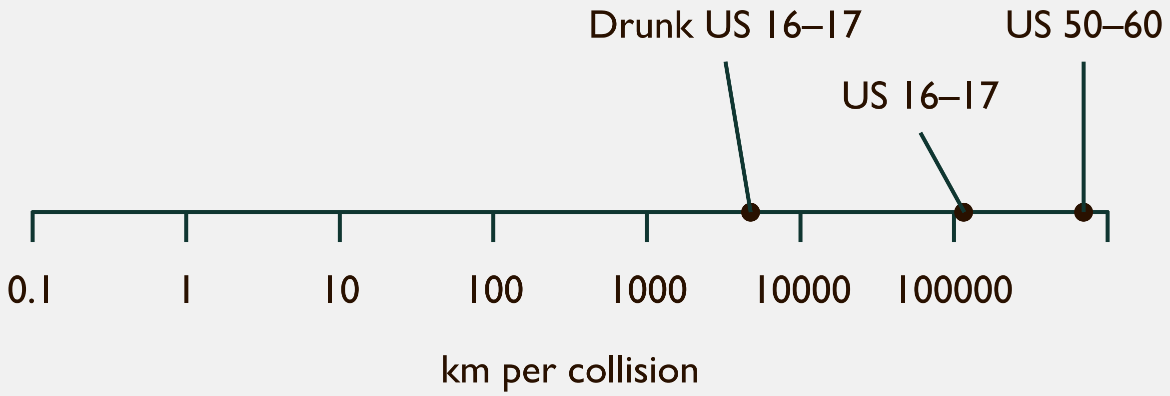
FIND UNDESIRED RISK TOLERANCE VIA INDIFFERENCE POINTS



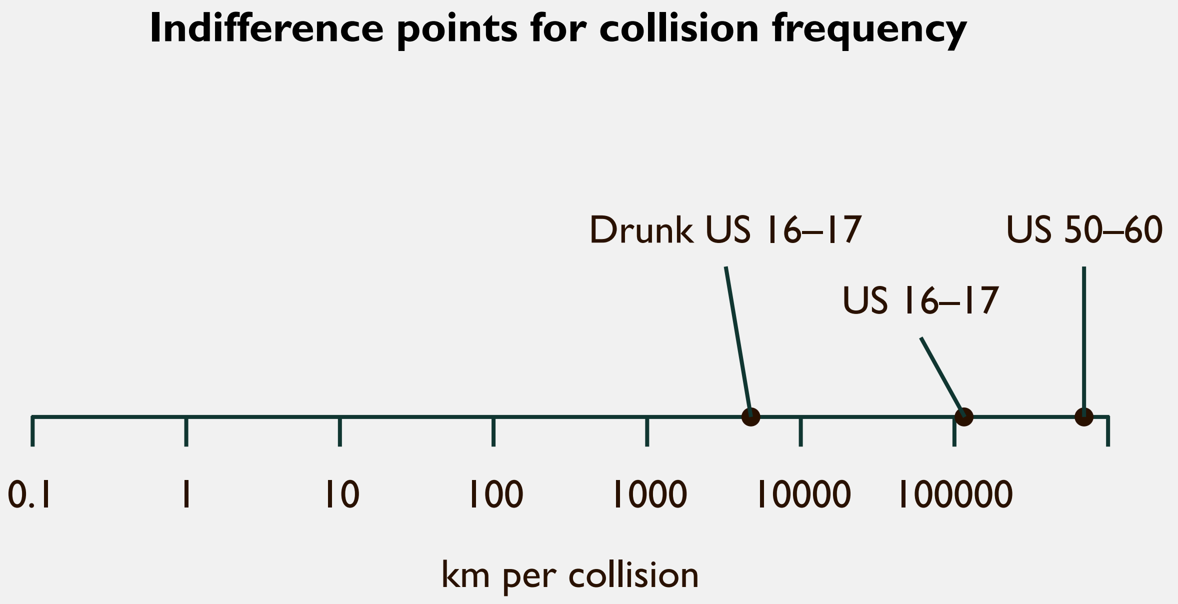
FIND UNDESIRED RISK TOLERANCE VIA INDIFFERENCE POINTS



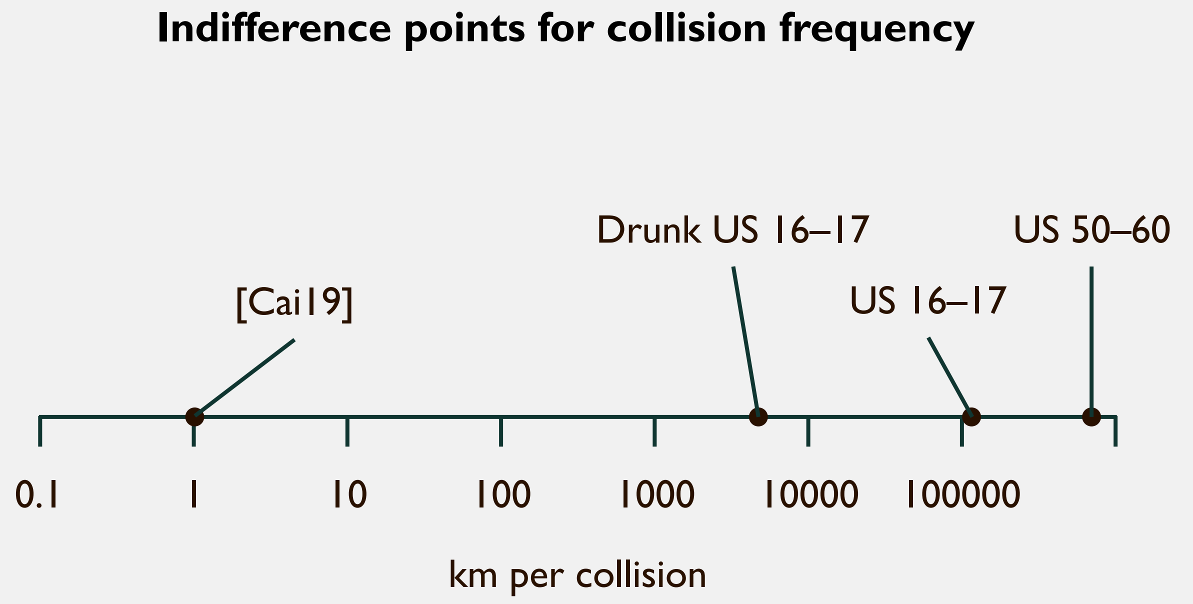
FIND UNDESIRED RISK TOLERANCE VIA INDIFFERENCE POINTS



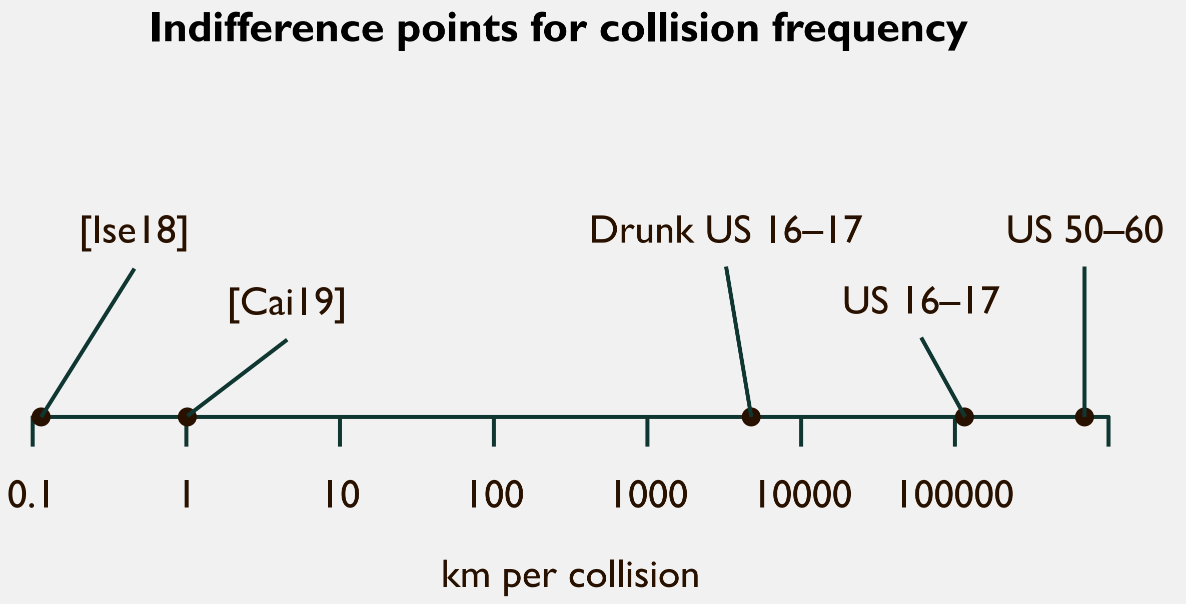
FIND UNDESIRED RISK TOLERANCE VIA INDIFFERENCE POINTS



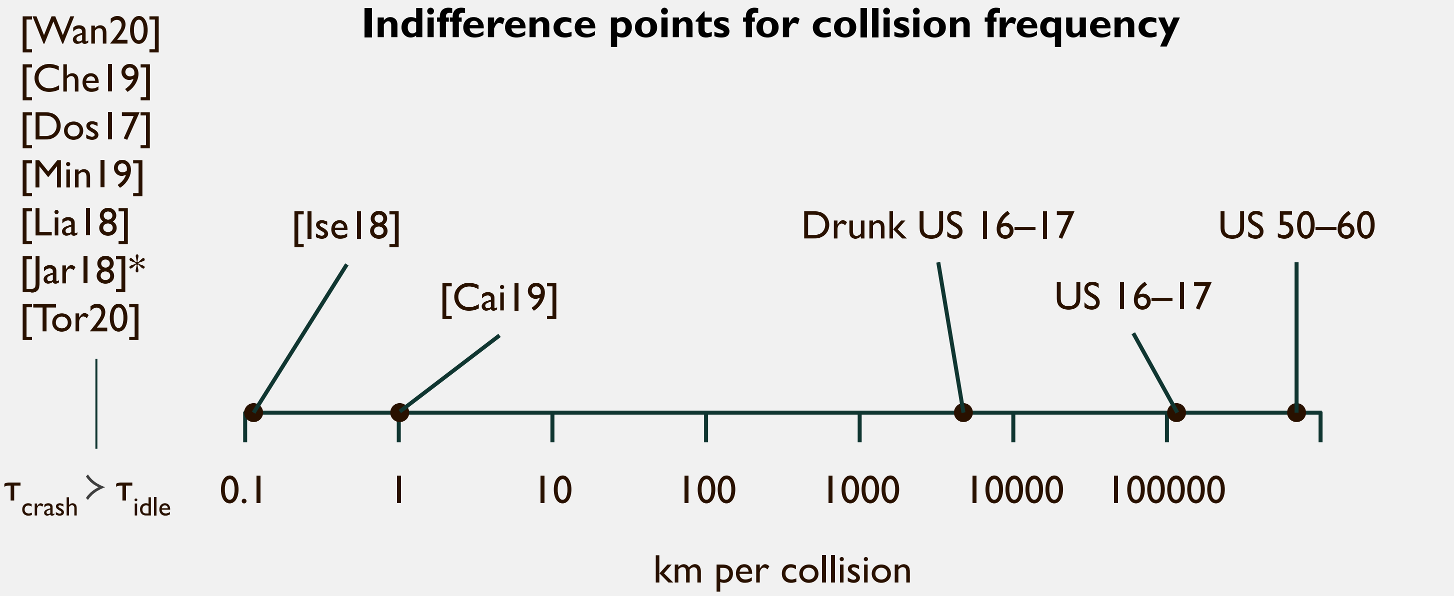
FIND UNDESIRED RISK TOLERANCE VIA INDIFFERENCE POINTS



FIND UNDESIRED RISK TOLERANCE VIA INDIFFERENCE POINTS



FIND UNDESIRED RISK TOLERANCE VIA INDIFFERENCE POINTS



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 ⟶ 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. 😱

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
  - 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

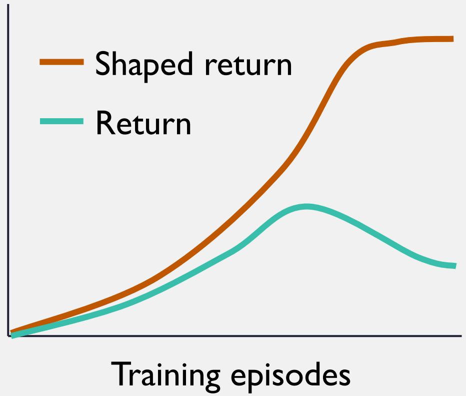
Reward shaping in RL for AD papers

- 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)**



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?

A trial-and-error process

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

A trial-and-error process

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

Step 2: Pick an RL algorithm for testing.

A trial-and-error process

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

Step 2: Pick an RL algorithm for testing.

Step 3: Learn a behavior policy.

A trial-and-error process

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.

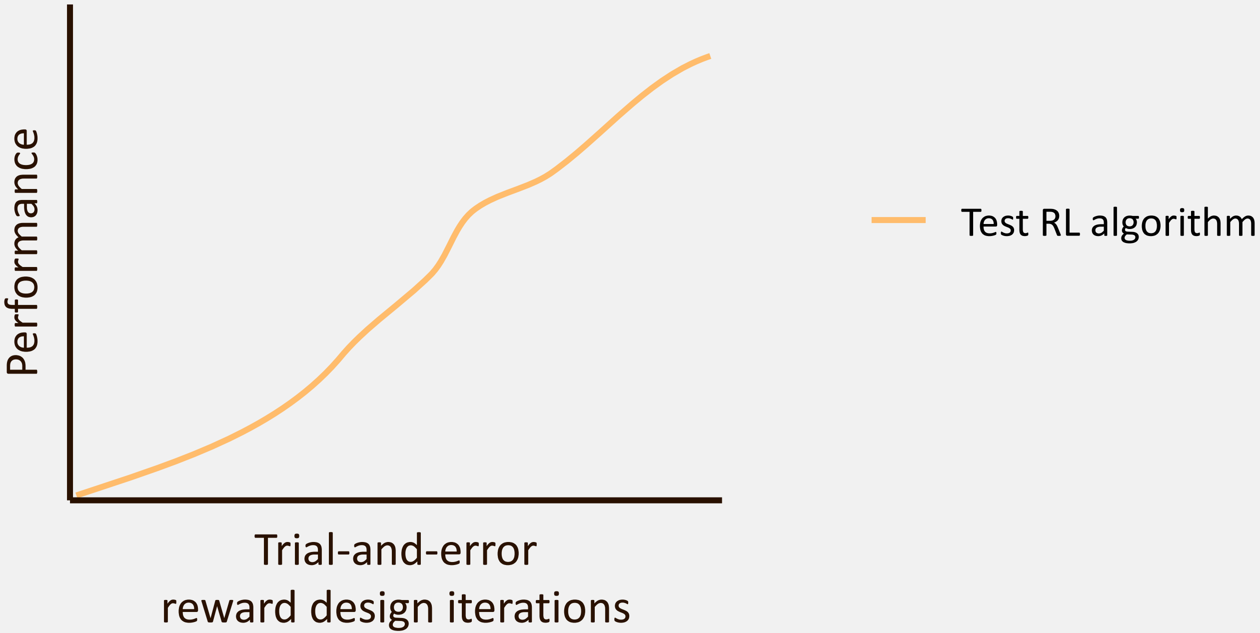
RL for AD

Of 8 papers whose authors shared their reward design process over email,
100% used trial-and-error to design their reward function.

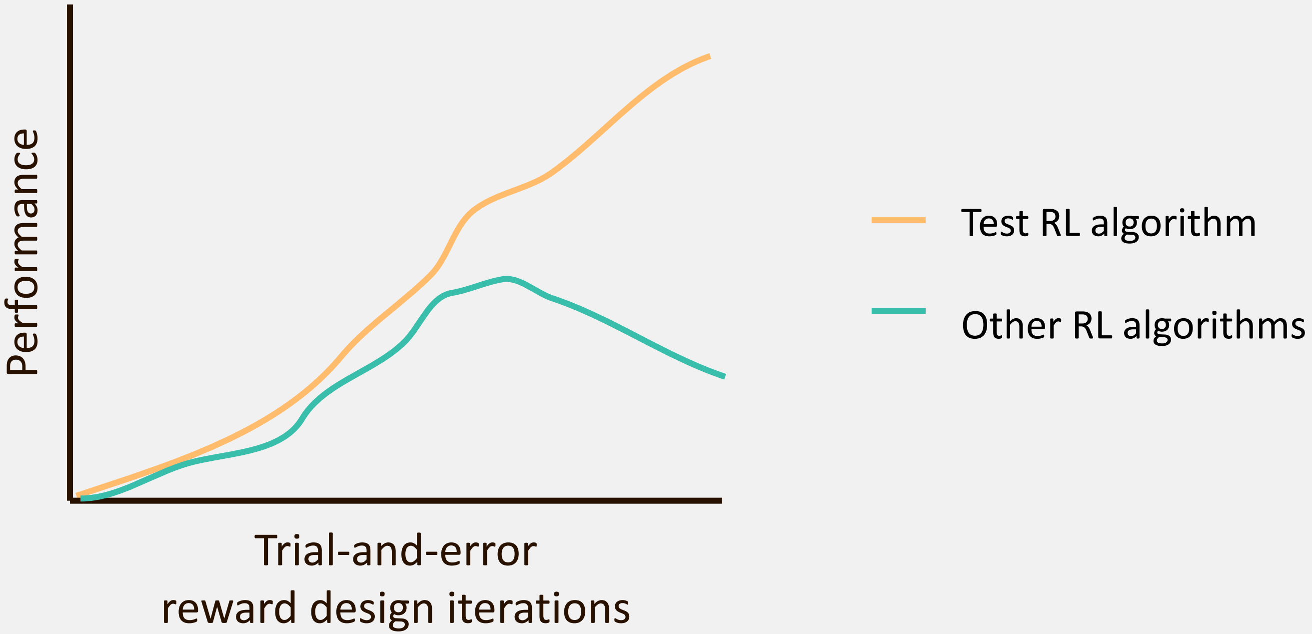
General RL experts

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

Overfitting the reward function to the training context?

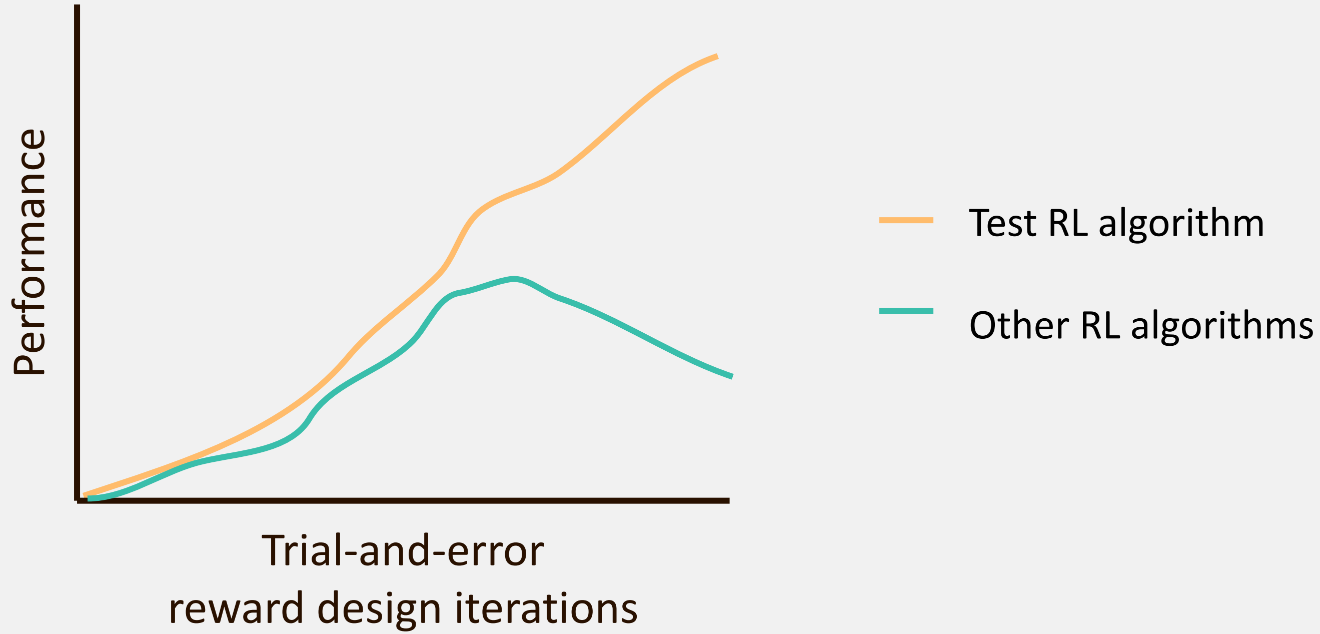


Overfitting the reward function to the training context?



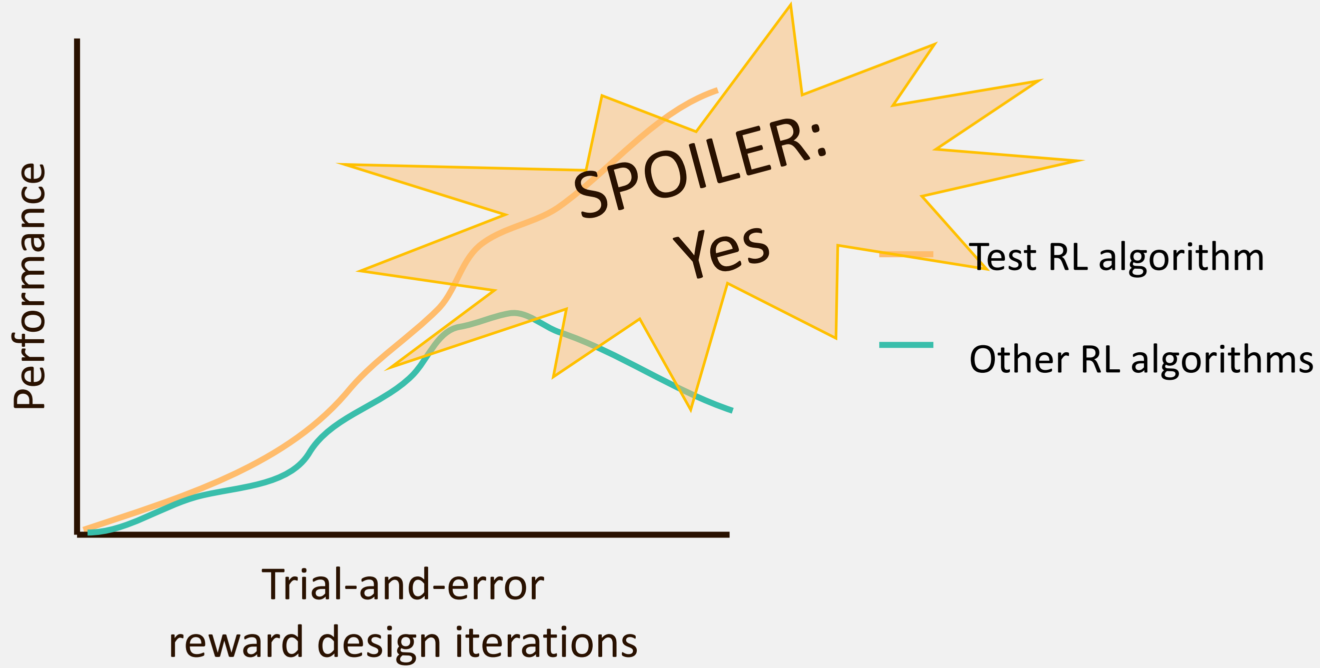
Overfitting the reward function to the training context?

Training context - RL algorithm, hyperparameters, and tasks

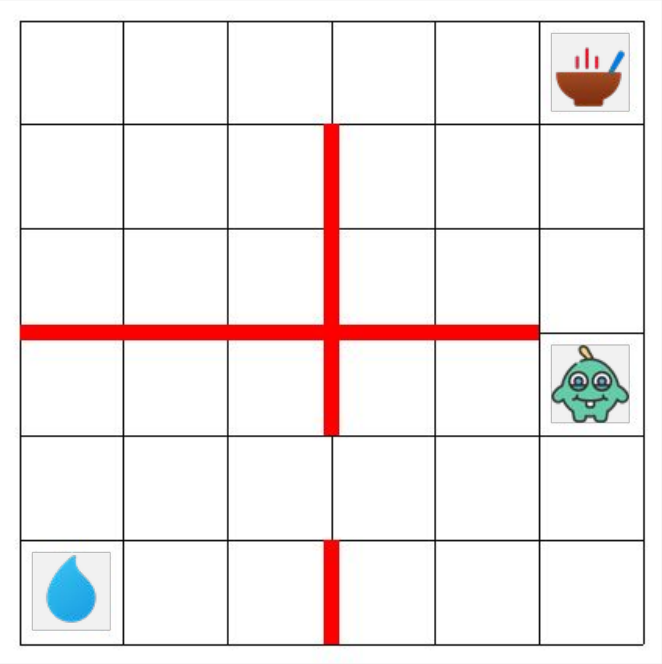


Overfitting the reward function to the training context?

Training context - RL algorithm, hyperparameters, and tasks

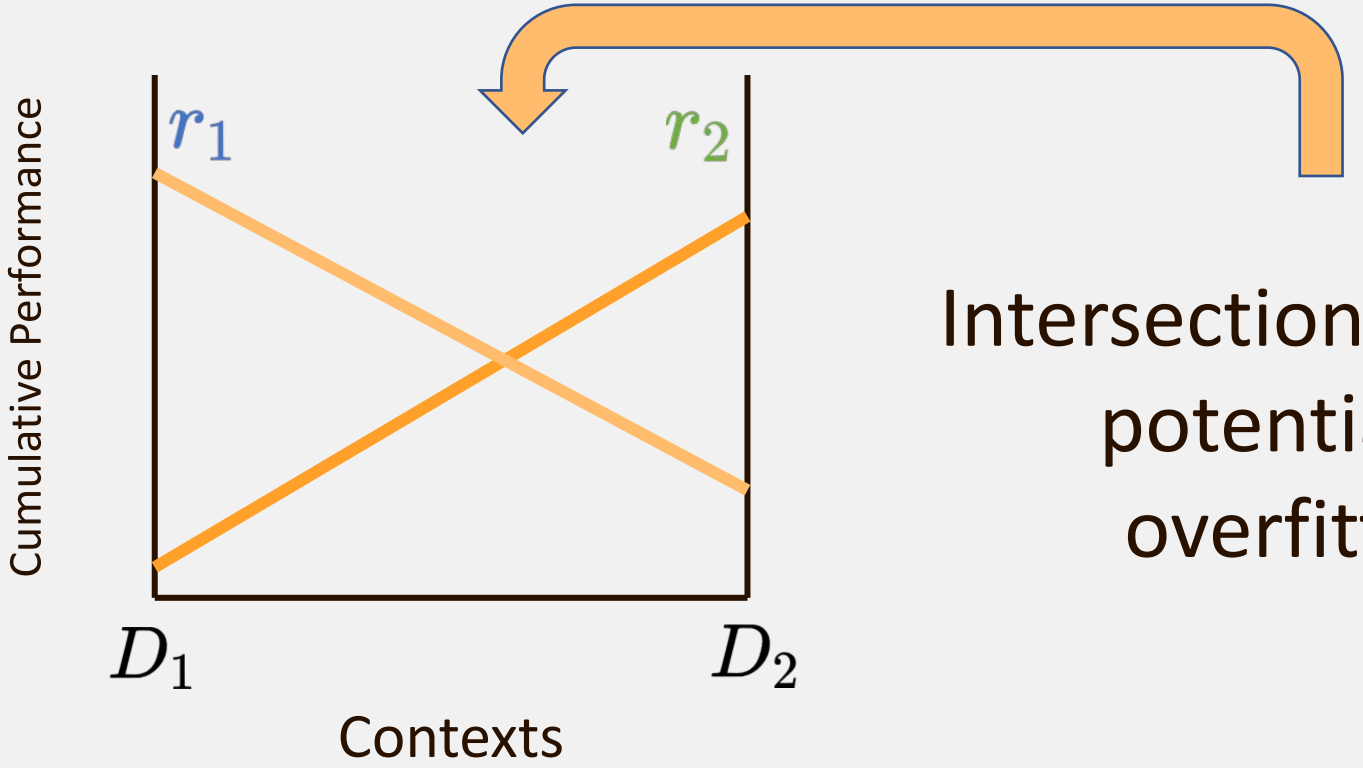


Hungry Thirsty Domain



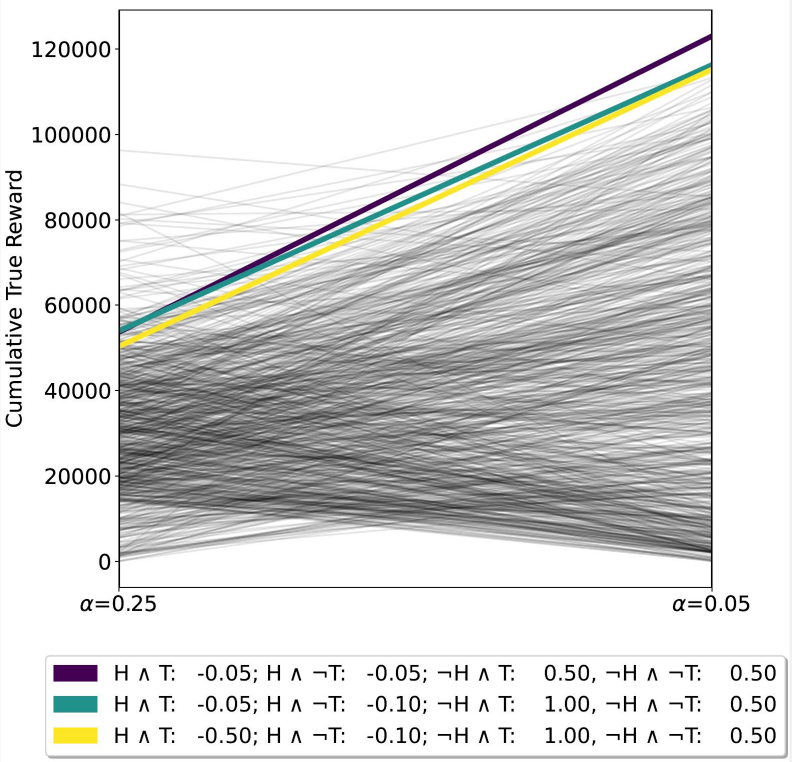
Singh et al., 2009, Where Do Rewards Come From?

Finding the potential for overfitting.



Intersections indicate potential for overfitting.

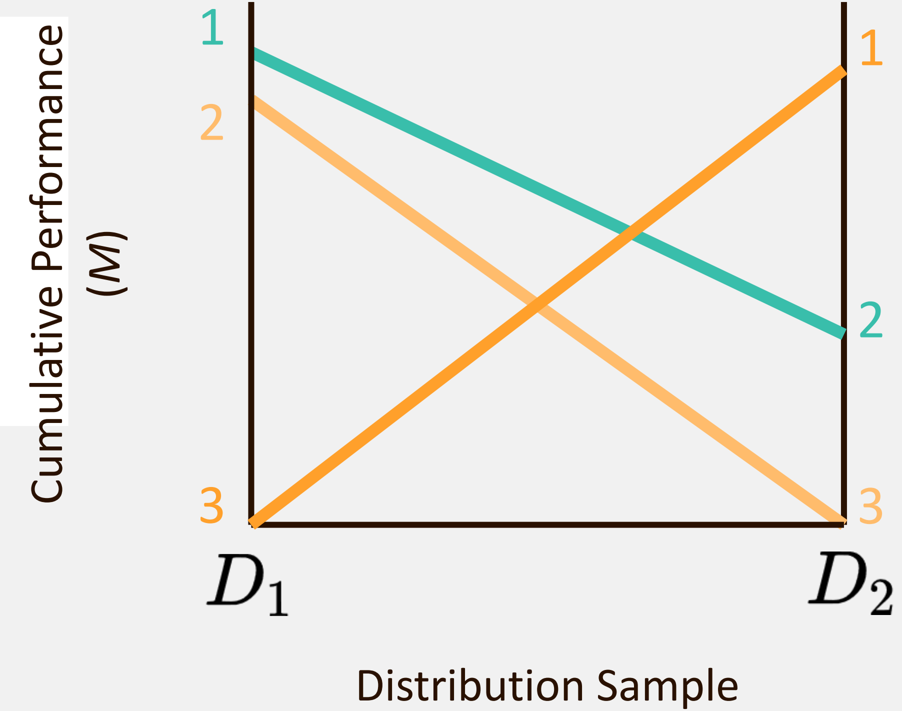
Finding: **Reward functions that achieve the best performance in one learning context can be suboptimal in another.**



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_1 & D_2)

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

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

$\mathcal{D}_1$	$\mathcal{D}_2$	τ_b
$\gamma = 0.99$	$\gamma = 0.8$	0.07
$\gamma = 0.99$	$\gamma = 0.5$	0.04
$\gamma = 0.8$	$\gamma = 0.5$	0.12
$\alpha = 0.25$	$\alpha = 0.05$	0.11

We rank all reward functions for each experiment setting ($\mathcal{D}_1$ & $\mathcal{D}_2$

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

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

User Study Conducted in Jupyter Notebooks

Reward for state: hungry AND thirsty -0.50

Reward for state: hungry AND not thirsty -0.25

Reward for state: not hungry AND thirsty 0.75

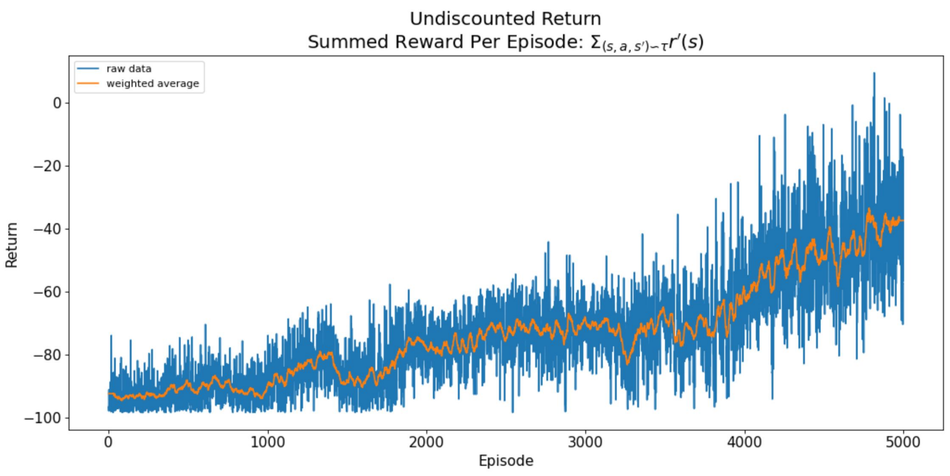
Reward for state: not hungry AND not thirsty 1.00

Algorithm Choice: DDQN

gamma: 0.99

num_episodes: 5000

lr: 0.001



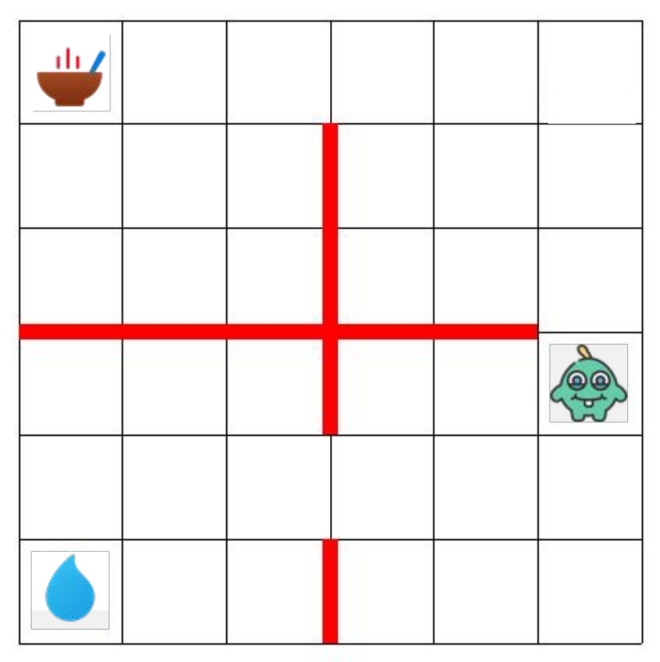
Experts overfit reward functions too

68% of users overfit reward functions

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

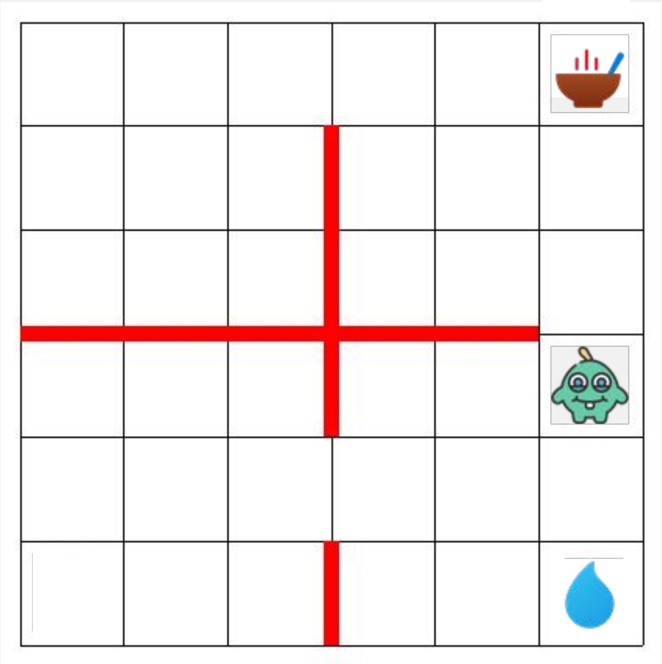
They ultimately selected a different reward function, which achieved a mean score of 1,031 with DDQN.

Experts' reward functions tend to not generalize.



Hard configuration

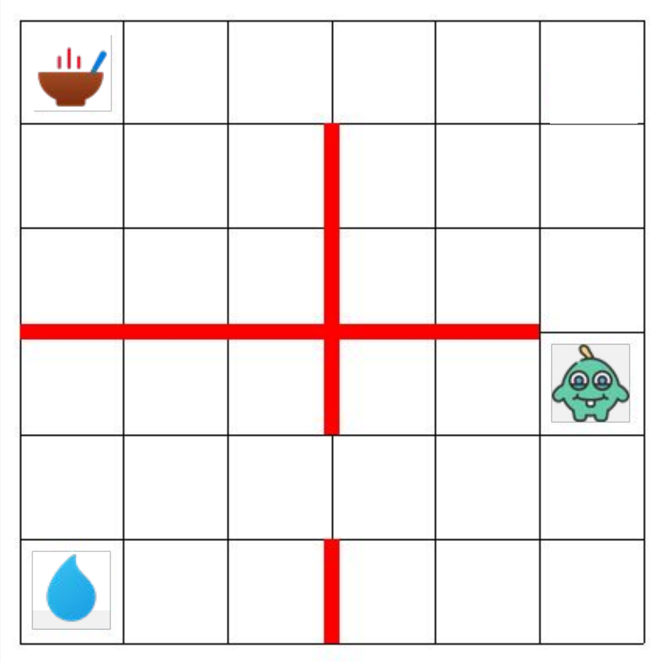
(15 steps between water & food)



Easy configuration

(5 steps between water & food)

Experts' reward functions tend to not generalize.

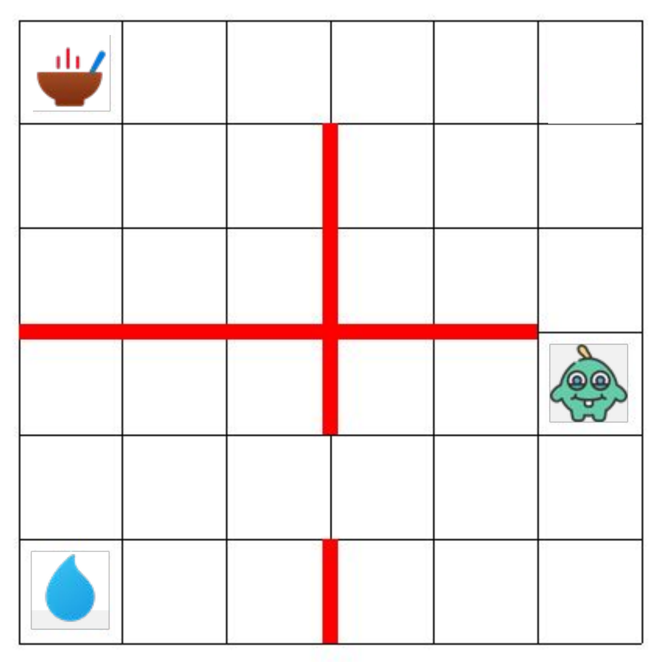


Hard configuration

(15 steps between water & food)

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

Experts are currently bad at writing reward functions.



Hard configuration

(15 steps between water & food)

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

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

$$r(\neg\mathtt{H} \wedge \neg\mathtt{T}) = 1.0 \qquad r(\mathtt{H} \wedge \neg\mathtt{T}) = -0.1$$
$$r(\neg\mathtt{H} \wedge \mathtt{T}) = 1.0 \qquad r(\mathtt{H} \wedge \mathtt{T}) = -1.0$$

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

Takeaways

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

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

Field		
reinf. learning	return	reward
motion planning	-1 × cost	-1 × cost
control theory	-1 × cost	-1 × cost
evolutionary algs.	fitness	-
utility theory	utility	-
optimization	objective function*	-
-	performance metric	-
-	score	-

* "objective function" more precisely refers to the expectation of G(τ)

REWARD AND RETURN

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

REWARD AND RETURN

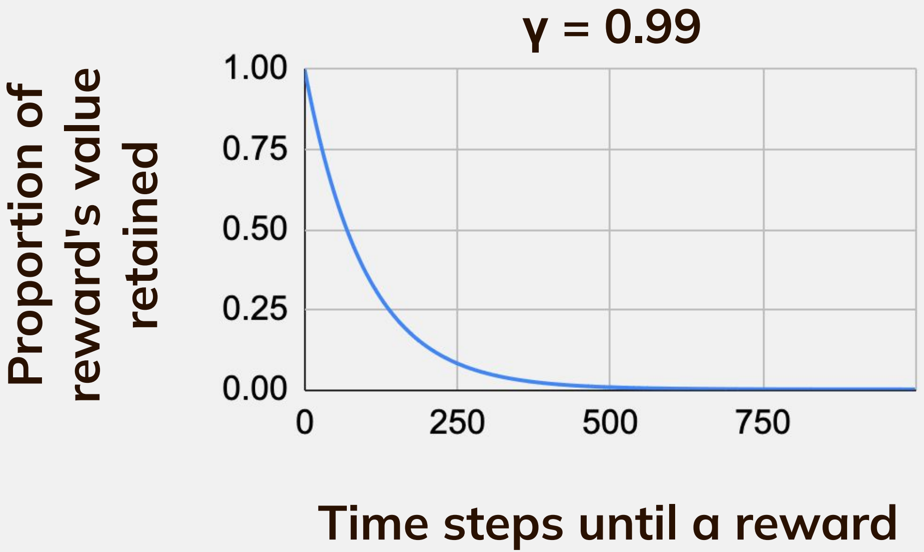
$$G(\tau) = \sum_{t=0}^{T} \gamma^t R(s_t, a_t, s_{t+1})$$

discount factor

REWARD AND RETURN

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

discount factor



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

Problem-side, γ_{MDP} - 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_{alg} - a hyperparameter of the RL algorithm

- $\boldsymbol{\gamma_{alg}} \leq \boldsymbol{\gamma_{MDP}}$
- $\boldsymbol{\gamma_{alg}} \leq 0.999$ in deep RL papers I have seen, usually $\boldsymbol{\gamma_{alg}} \leq 0.99$
- in practice, $\boldsymbol{\gamma_{alg}}$ 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_{MDP} unless otherwise stated.

Contemporary RL tends to have <u>2 discount factors</u>: problem-side and algorithmic

Problem-side, γ_{MDP} - part of the RL problem definition

- creates the true return

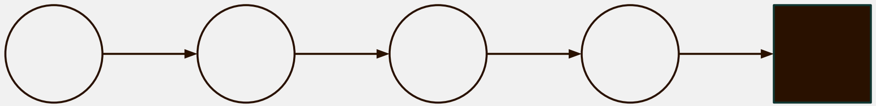
Algorithmic, γ_{alg} - a hyperparameter of the RL algorithm

- $\gamma_{alg} \leq \gamma_{MDP}$
- $\gamma_{alg} \leq 0.999$ in deep RL papers I have seen, usually $\gamma_{alg} \leq 0.99$
- in practice, γ_{alg} 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_{MDP} unless otherwise stated.

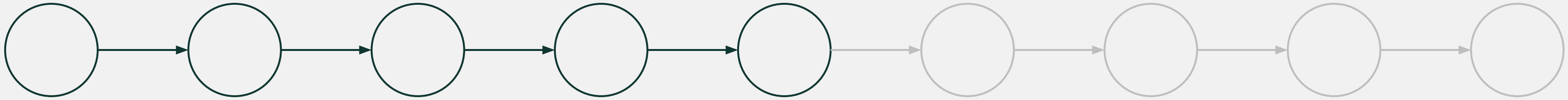
Estimating return during RL at absorbing state vs. when stopping an episode for other reasons

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

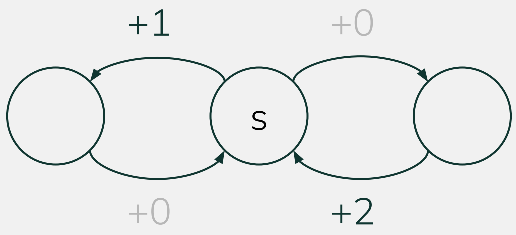


When function approximation is used, there is danger that value inference will return a nonzero value. You can use γ=0 to get the equivalent effect as having a value of 0.

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



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



In this continuing domain,

- if γ < 0.5, then choosing the left loop from s is optimal
- if γ > 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 γ=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 $\log_\gamma$

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 $= \log_\gamma 0.1 * 0.1s$
- 1% - 4.37 s $= \log_\gamma 0.01 * 0.1s$
- 0.1% - 6.56 s $= \log_\gamma 0.001 * 0.1s$

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 $\log_\gamma$

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:

- 10% - 22.9s $= \log_\gamma 0.1 * 0.1s$
- 1% - 45.8s $= \log_\gamma 0.01 * 0.1s$
- 0.1% - 68.7s $= \log_\gamma 0.001 * 0.1s$

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 $\log_\gamma$

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 - γ) results in a ~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.

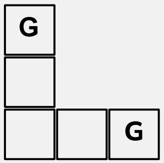
It's not necessary though in episodic tasks, so to avoid these drawbacks it should not be used.

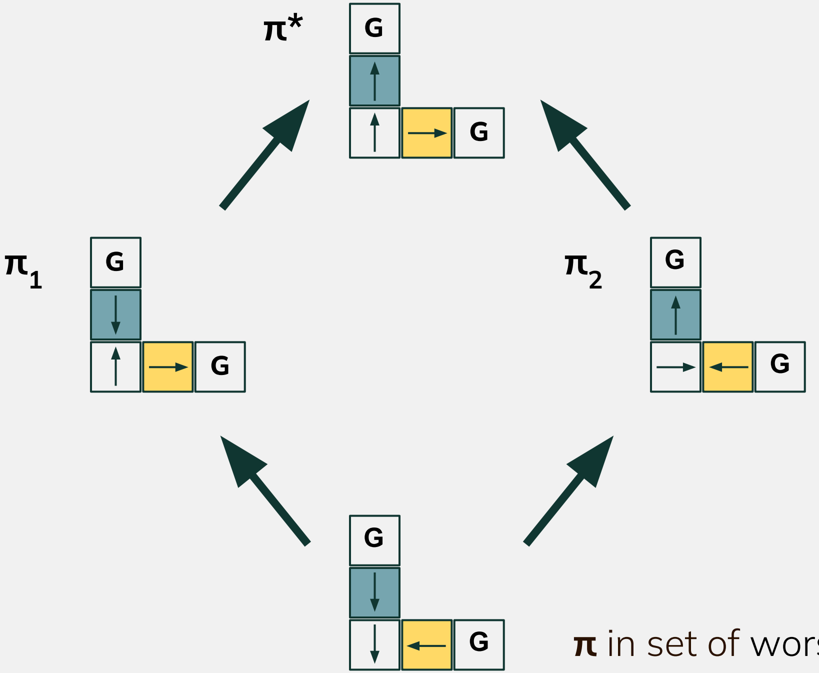
Whether you use discounting for your *algorithm* is a different matter.

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

MDP

Reward:
-1 per time step





$\boldsymbol{\pi}$ in set of worst πs

If the start state is the **yellow cell**, $\boldsymbol{\pi_1}$ has higher expected return.

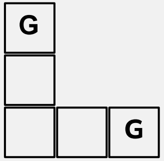
If the start state is the **aqua cell**, $\boldsymbol{\pi_2}$ has higher expected return.

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

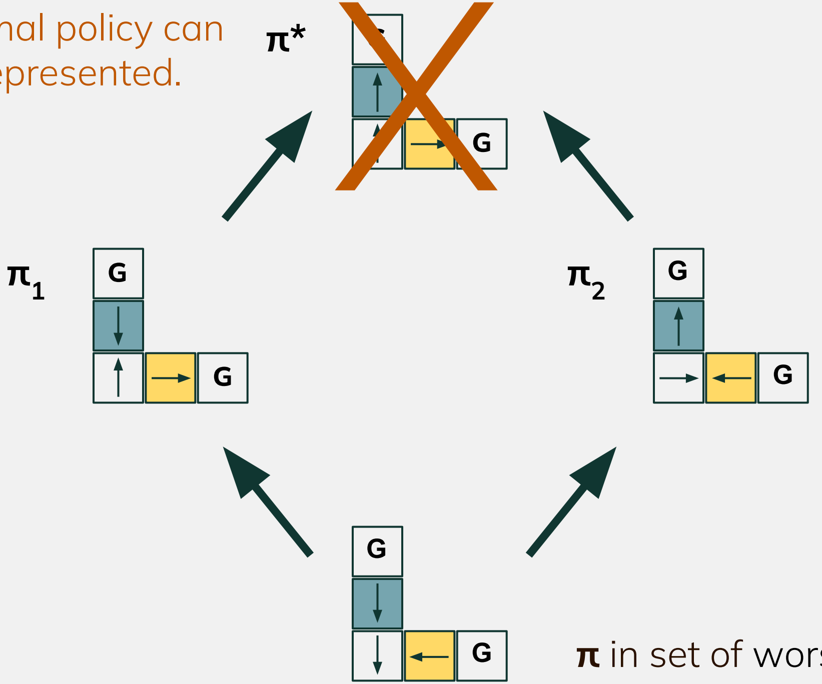
A continuing exponentially discounted task may not have an optimal policy under function approximation.

MDP

Reward:
-1 per time step



It's plausible that no optimal policy can be represented.



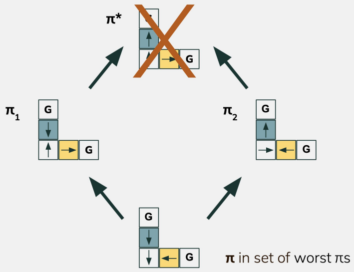
If the start state is the **yellow cell**, π_1 has higher expected return.

If the start state is the **aqua cell**, π_2 has higher expected return.

π in set of worst πs

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 **$v^{\pi}(s) \geq v^{\pi'}(s)$** ***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.

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

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 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.

Methods for finding misalignment become methods for optimizing the reward function via RLHF.

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

Summary

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. **Reward (Mis)design for Autonomous Driving**. AIJ 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>