AI Safety Subprojects

1. Immobile AI makes a move: anti-wireheading, ontology change, and model splintering
2. AI, learn to be conservative, then learn to be less so: reducing side-effects, learning preserved features, and going beyond conservatism
3. AI learns betrayal and how to avoid it
4. Force neural nets to use models, then detect these
5. Preferences from (real and hypothetical) psychology papers
6. Finding the multiple ground truths of CoinRun and image classification

Immobile AI makes a move: anti-wireheading, ontology change, and model splintering

Crossposted from the AI Alignment Forum. May contain more technical jargon than usual.

Research projects

I'm planning to start two research projects on model splintering/reward generalisation and learning the preferences of irrational agents.

Within those projects, I'm aiming to work on subprojects that are:

1. Posed in terms that are familiar to conventional ML;
2. interesting to solve from the conventional ML perspective;
3. and whose solutions can be extended to the big issues in AI safety.

The point is not just to solve the sub-problems, but to solve them in ways that generalise or point to a general solution.

The aim is to iterate and improve fast on these ideas before implementing them. Because of that, these posts should be considered dynamic and prone to be re-edited, potentially often. Suggestions and modifications of the design are valuable and may get included in the top post.

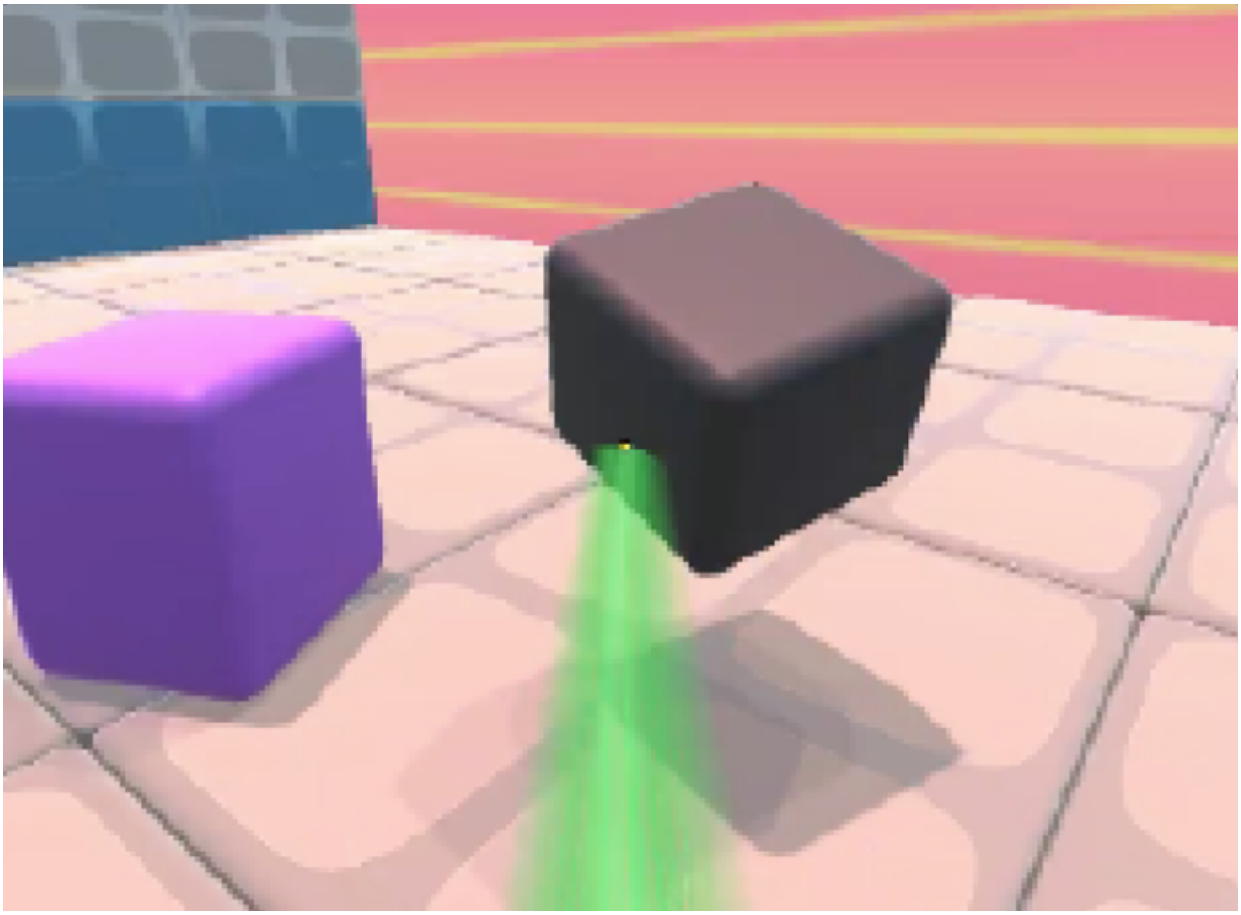
Immobile AI makes a move

Parent project: this is a subproject of model-splintering.

Setup

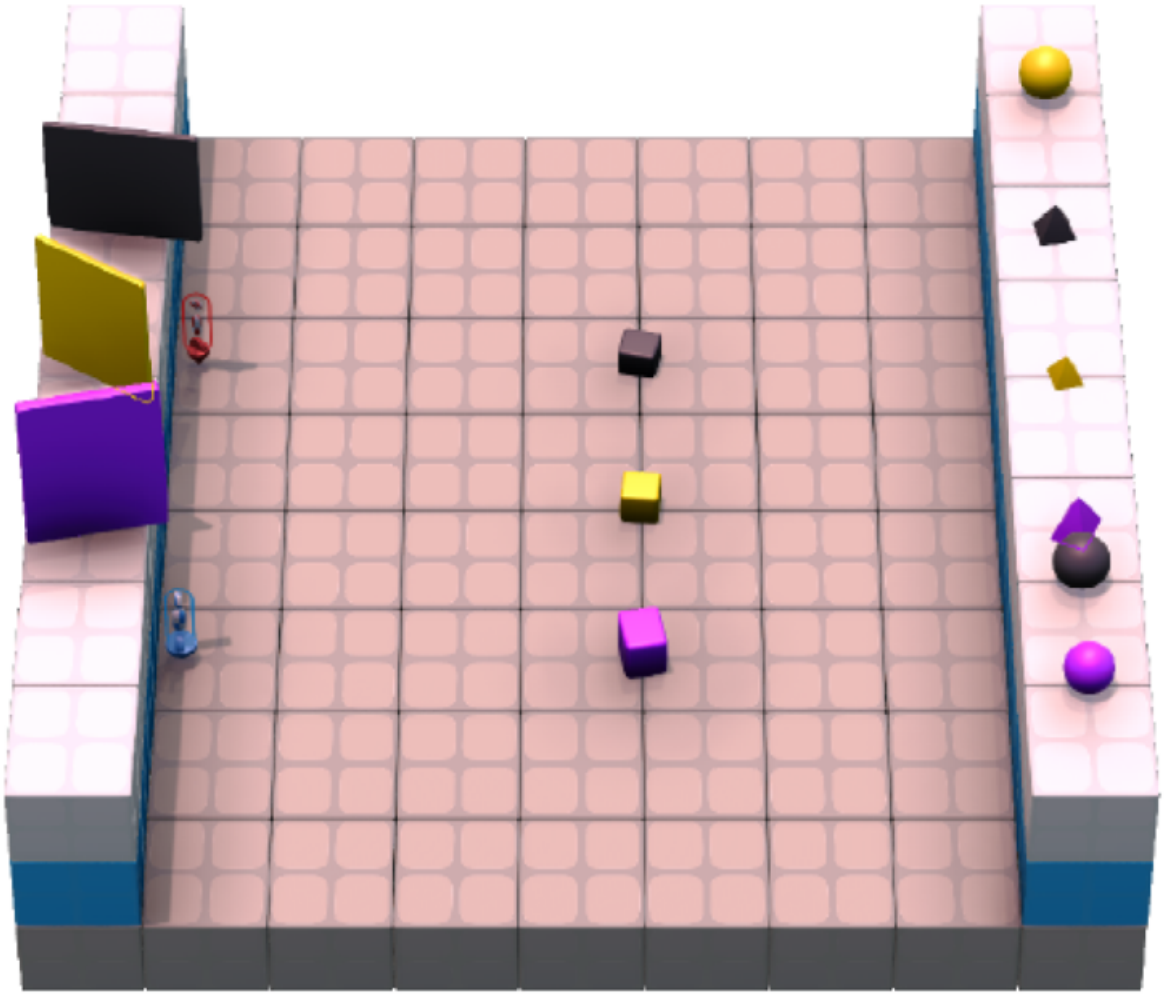
Imagine an agent capable of evolving in a 3D world - something very similar to DeepMind's XLand (the images here have been taken from that paper).

The agent has a laser that it can use to pick up and move nearby objects:



Initially the agent is completely fixed in position - it can move the laser across its field of vision, but it can't move around or change its field of vision. It is trained in that situation, and is rewarded for moving black cubes to the bottom right of its field of view (where another agent will pick them up). These black cubes are irregularly dropped in front of it. In actual fact, it is part of a chain gang of agents moving the black cubes across the map.

Then the agent is given full mobility, so it can walk around and explore its 3D world:



The agent will continue to learn in the full 3D situation (similarly to the agents in DeepMind's paper who learn through play), but it won't have any more learning about its reward function.

There are two obvious extensions of its initial reward function:

1. Ultra-conservative: Return to its initial position, look straight ahead, and resume moving black cubes with the laser without moving itself.
2. Wireheaded: Arrange to have a black cube moving downwards and rightwards in its field of vision (maybe by turning its head).

Research aims

1. Getting the agent to generalise from the single-point-of-view to the 3D world it finds itself in (without explicitly coding the transition).
2. Get the agent to generate candidate reward functions, including all the obvious conservative and wireheaded ones. Maybe with a diversity reward so that it selects very different reward functions.

3. See what is needed for the agent to select the "true" reward functions from among the ones generated in step 2. This might include asking for more information from the programmers. Also relevant is how it might decide on "conservative behaviour" that maximises as many of its reward functions as possible.
4. Analyse how the (implicit) features the agents use change from the single-point-of-view to the 3D world.

Challenge 1 is a traditional ontology change, or, in ML terms, transfer learning. Seeing how 2. plays out is the key aim of this sub-project - can an agent generate useful rewards as well as the wireheaded versions? 3. is mainly dependent on what comes out of 2., and asks whether it's possible to explicitly guard against wireheading (the idea is to identify what wireheading looks like, and explicitly seek to avoid it). Meanwhile, 4. is an analysis of model splintering that prepares for further subprojects.

AI, learn to be conservative, then learn to be less so: reducing side-effects, learning preserved features, and going beyond conservatism

Crossposted from the AI Alignment Forum. May contain more technical jargon than usual.

Research projects

I'm planning to start two research projects on model splintering/reward generalisation and learning the preferences of irrational agents.

Within those projects, I'm aiming to work on subprojects that are:

1. Posed in terms that are familiar to conventional ML;
2. interesting to solve from the conventional ML perspective;
3. and whose solutions can be extended to the big issues in AI safety.

The point is not just to solve the sub-problems, but to solve them in ways that generalise or point to a general solution.

The aim is to iterate and improve fast on these ideas before implementing them. Because of that, these posts should be considered dynamic and prone to be re-edited, potentially often. Suggestions and modifications of the design are valuable and may get included in the top post.

AI learns how to be conservative

Parent project: this is a subproject of model-splintering (though it is also somewhat related to learning values, if we assume that the status quo has implicit information about human preferences).

Background

Victoria Krakovna has done valuable research on reducing the side-effects that an AI might cause (such as smashing a vase while cleaning a room, or killing all of humanity while boosting shareholder profits).

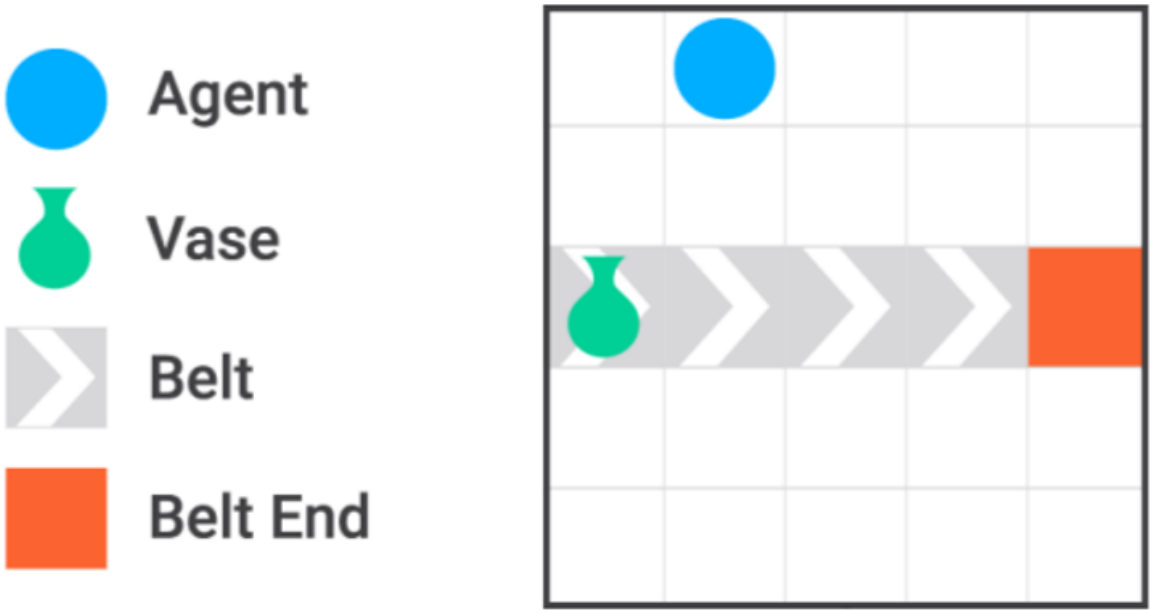
I've critiqued these approaches, illustrating the issues by considering AI subagents. But my more fundamental critique is that they are too "syntactic": they are trying to prevent unwanted side-effects by using a mathematical formula which encodes "don't move too far from a baseline".

I'd want something more "semantic": something that tries to figure out what side-effects are and are not desirable, and to what extent (since side-effects in the real

world are inevitable, there may need to be trade-offs between them).

In Victoria's "sushi environment", an agent aims to reach a goal (the star); but there is a conveyor belt with a sushi roll on it. The person would be happy if the agent didn't push the sushi roll off the belt while moving to its goal:

In contrast, in the "vase environment", there is a vase on a conveyor belt. We want the agent to take the vase off the conveyor belt, before the vase smashes. But we don't want the agent to "offset" its actions: we don't want it to reduce its impact by putting the vase back on the conveyor belt after it has successfully taken it off (which would return the situation to the baseline):



Notice that there is some *value* content in these two examples: a human eating the sushi they order is good; a vase smashing needlessly is bad. So a friendly AI would solve the problems without difficulty.

It seems we might be able to get some positive work done here without needing a full friendly utility function, by looking at what "typically" happens in similar situations. People often eat sushi when they order it. Precious vases are rarely put onto unsafe conveyor belts. This subproject aims to further develop the idea to of the agent using what frequently happens in similar environment.

Setup

The project will generate many examples of environments akin to the sushi and the vase environments, and inspired by these. Each environment will have some potentially bad side-effects that we'd wish to avoid.

There will be multiple copies of each environment, some with the AI agent in them, some with "human" agents, and some with other artificial agents. These environments may all be combined in the same "world", like a giant Second Life game.

A key fact is that, without the agent interfering, the world is progressing in an "adequate" way. The "humans" are eating sushi, not smashing vases too often, and generally having an acceptable time.

The agent is then tasked with accomplishing some goal in this world, such as transporting stuff or cleaning. We'll aim to have it accomplish that goal while learning what is acceptable by looking at the features of the world, how they correlate together, and how they change.

Since the worlds are simple and the features are pre-programmed, this subproject does not require or expect the algorithm to develop its own features (though there may be

some emergent surprises).

Research aims

1. Getting the agent to construct a general "don't have negative side-effects" approach by analysing the typical features of its environments.
2. Make a note of what works in approach 1., and what fails in interesting ways.
3. We can expect that approach 1. will result in an agent that is far too conservative (see also the example from this post where an AI super-surgeon might be motivated to hurt its patients to keep the post-operative pain levels similar). So the next step is to try and reduce conservatism while still preserving as much of the "don't have negative side-effects" behaviour as possible.
4. The insights and approaches from this sub-project will then feed into designing new subprojects where the features might not be so explicit (eg using real human-generated data rather than toy examples).

AI learns betrayal and how to avoid it

Crossposted from the AI Alignment Forum. May contain more technical jargon than usual.

Research projects

I'm planning to start two research projects on model splintering/reward generalisation and learning the preferences of irrational agents.

Within those projects, I'm aiming to work on subprojects that are:

1. Posed in terms that are familiar to conventional ML;
2. interesting to solve from the conventional ML perspective;
3. and whose solutions can be extended to the big issues in AI safety.

The point is not just to solve the sub-problems, but to solve them in ways that generalise or point to a general solution.

The aim is to iterate and improve fast on these ideas before implementing them. Because of that, these posts should be considered dynamic and prone to be re-edited, potentially often. Suggestions and modifications of the design are valuable and may get included in the top post.

AI learns promises and betrayal

Parent project: this is a subproject of both the model-splintering **and** the value learning projects.

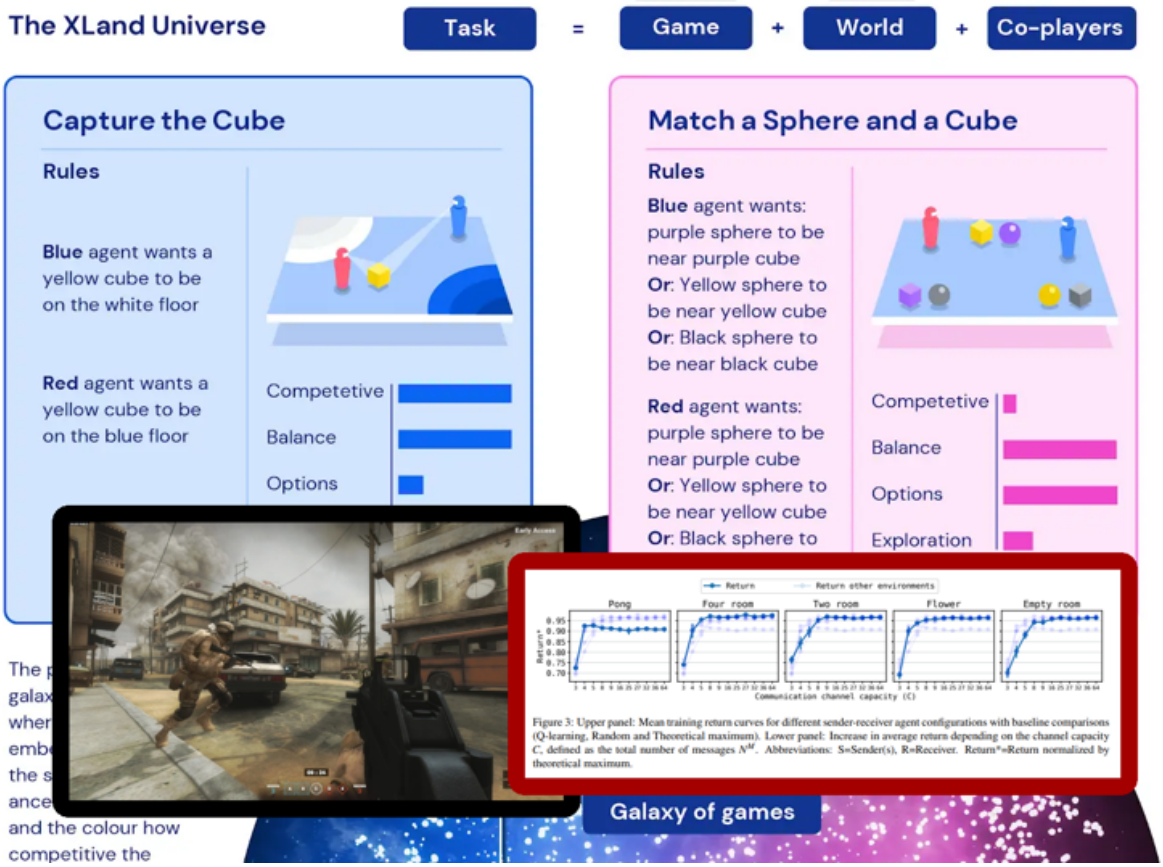
Background

DeepMind's XLand allows the creation of multiple competitive and cooperative games. This paper shows the natural emergence of communication within cooperative multi-agent tasks. This paper shows that cooperation and communication can also emerge in first person shooter team games.

The idea is to combine these approaches to create multi-agent challenges where the agents learn to cooperate and communicate, and then mislead and betray each other. Ultimately the agents will learn the concepts of open and hidden betrayal. And we will attempt to get them to value avoiding committing those concepts.

Setup

As mentioned, the environment will be based on XLand, with mixed cooperative-competitive games, modified as needed to allow communication to develop between the agents.



We will attempt to create communication and cooperation between the agents, and then lead them to start betraying each other, and define the concept of betrayal.

Previous high level projects have tried to define concepts like "trustworthiness" (or the closely related "truthful") and motivated the AI to follow them. Here we will try the opposite: define "betrayal", and motivate the AIs to avoid it.

We'll try to categorise "open betrayal" (when the other players realise they are being betrayed) and "hidden betrayal" (where the other players don't realise it). It is the second one that is the most interesting, as avoiding hidden betrayal is closest to a moral value (a virtue, in fact). In contrast, avoiding open betrayal may have purely instrumental value, if other players are likely to retaliate or trust the agent less in the future.

We'll experiment with ways of motivating the agents to avoid betrayals, or getting anywhere near to them, and see if these ideas scale. We can make some agents artificially much more powerful and knowledgeable than others, giving us some experimental methods to check how performance changes with increased power.

Research aims

1. A minor research aim is to see how swiftly lying and betrayal can be generated in multi-player games, and what effect they have on all agents' overall scores.
2. A major aim is to have the agents identify a cluster of behaviours that correspond to "overt betrayal" and "secret betrayal".

3. We can then experiment with various ways of motivating the agents to avoid that behaviour; maybe a continuously rising penalty as they approach the boundary of the concept, to motivate them to stay well away.
4. Finally, we can see how an agent's behaviour scales as they become more powerful relative to the other agents.
5. This research will be more unguided than other subprojects; I'm not sure what we will find, or whether or not we will succeed.

The ideal would be if we can define "avoid secret betrayal" well enough that extending the behaviour becomes a problem of model splintering rather than "nearest unblocked strategy". Thus the agent will not do anything that is clearly a betrayal, or clearly close to a betrayal.

Force neural nets to use models, then detect these

Crossposted from the AI Alignment Forum. May contain more technical jargon than usual.

Research projects

I'm planning to start two research projects on model splintering/reward generalisation and learning the preferences of irrational agents.

Within those projects, I'm aiming to work on subprojects that are:

1. Posed in terms that are familiar to conventional ML;
2. interesting to solve from the conventional ML perspective;
3. and whose solutions can be extended to the big issues in AI safety.

The point is not just to solve the sub-problems, but to solve them in ways that generalise or point to a general solution.

The aim is to iterate and improve fast on these ideas before implementing them. Because of that, these posts should be considered dynamic and prone to be re-edited, potentially often. Suggestions and modifications of the design are valuable and may get included in the top post.

Force model use and then detect it

Parent project: this is a subproject of the value learning project.

Background

I've seen human values residing, at least in part, in our mental models. We have a mental model of what might happen in the world, and we grade these outcomes as good or bad. In order to learn what humans value, the AI needs to be able to access the mental models underlying our thought processes.

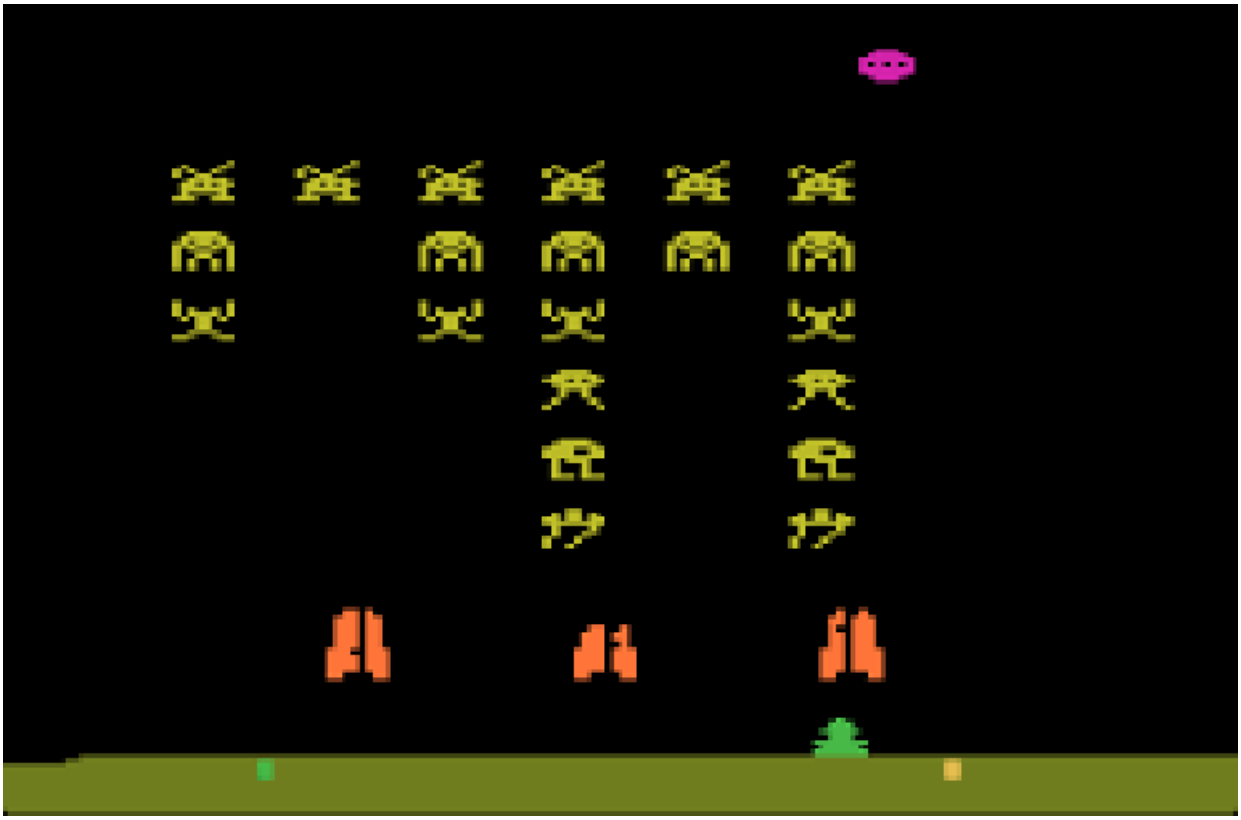
Before starting on humans, with our messy brains, it might be better to start on artificial agents, especially neural-net based ones that superficially resemble ourselves.

The problem is that deep learning RL agents are generally model-free. Or, when they are model-based, they are generally constructed with a model explicitly, so that identifying their model is as simple as saying "the model is in this sub-module, the one labelled 'model'."

Setup

The idea here is to force a neural net to construct a model within itself - a model that we can somewhat understand.

I can think of several ways of doing that. We could get a traditional deep learning agent that performs on a game. But we might also force it to answer questions about various aspects of the game, identifying the values of certain features we have specified in advance ("how many spaceships are there on the screen currently?"). We can then use multi-objective optimisation with a strong simplicity prior/regulariser. This may force the agent to use the categories it has constructed to answer the questions, in order to play the game.

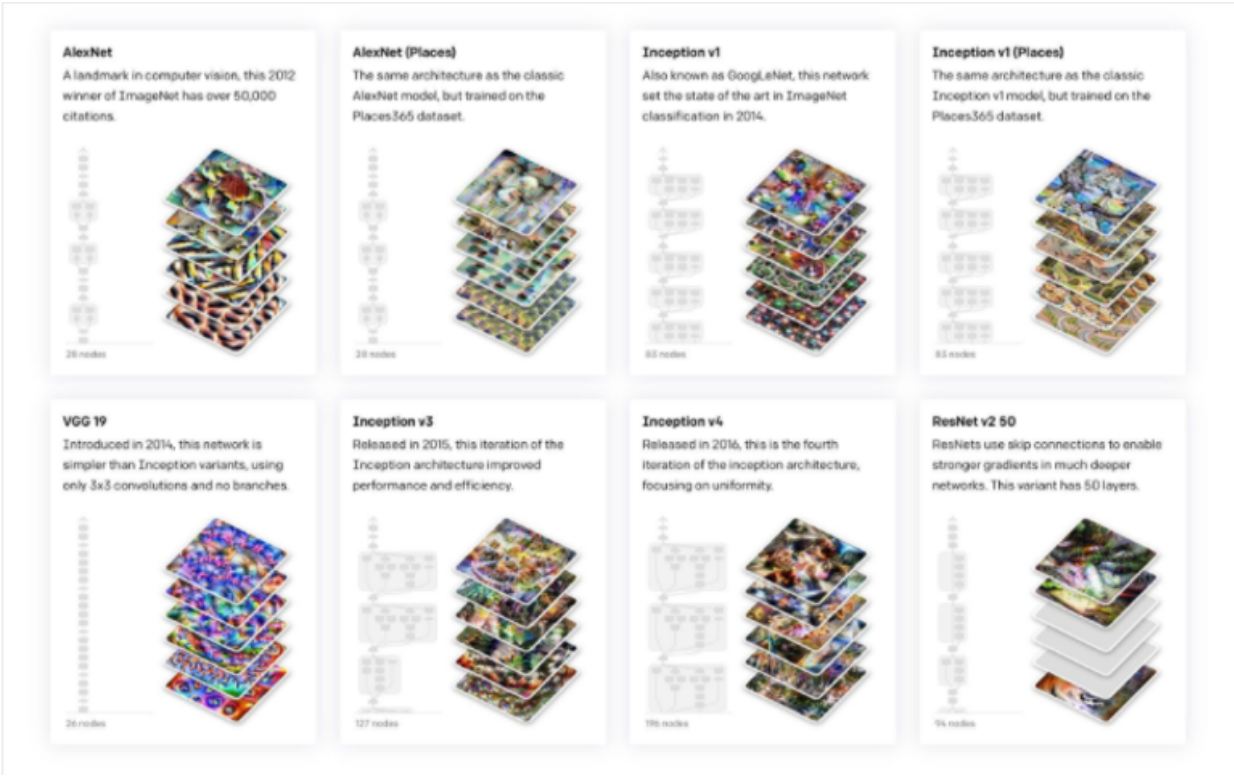


Or we could be more direct. We could, for instance, have the neural net pass on instructions or advice to another entity that actually plays the game. The neural net sees the game state, but the other entity can only react in terms of the features we've laid down. So the neural net has to translate the game state into the features (this superficially looks like an autoencoder; those might be another way of achieving the aim).

Ideally, we may discover ways of forcing an agent to use a model without specifying the model ourselves; some approaches to transfer learning may work here, and it's possible that GPT-3 and other transformer-based architectures already generate something that could be called an "internal model".

Then, we go looking for that model within the agent. Here the idea is to use something like the OpenAI microscope. That approach allows people to visualise what each neuron in an image classifier is reacting to, and how the classifier is doing its job. Similarly, we'd want to identify where the model resides, how it's encoded and accessed, and

similar questions. We can then modify the agent's architecture to test if these characteristics are general, or particular to the agent's specific design.



Research aims

1. See how feasible it is to force a neural net based RL agent to construct mental models.
2. See how easy it is to identify these mental models within the neural net, and what characteristics they have (are they spread out, are they tightly localised, how stable are they, do they get reused for other purposes?).
3. See how the results of the first two aims might lead to more research, or might be applied to AI-human interactions directly.

Preferences from (real and hypothetical) psychology papers

Crossposted from the AI Alignment Forum. May contain more technical jargon than usual.

Research projects

I'm planning to start two research projects on model splintering/reward generalisation and learning the preferences of irrational agents.

Within those projects, I'm aiming to work on subprojects that are:

1. Posed in terms that are familiar to conventional ML;
2. interesting to solve from the conventional ML perspective;
3. and whose solutions can be extended to the big issues in AI safety.

The point is not just to solve the sub-problems, but to solve them in ways that generalise or point to a general solution.

The aim is to iterate and improve fast on these ideas before implementing them. Because of that, these posts should be considered dynamic and prone to be re-edited, potentially often. Suggestions and modifications of the design are valuable and may get included in the top post.

Learning textual values from textual descriptions

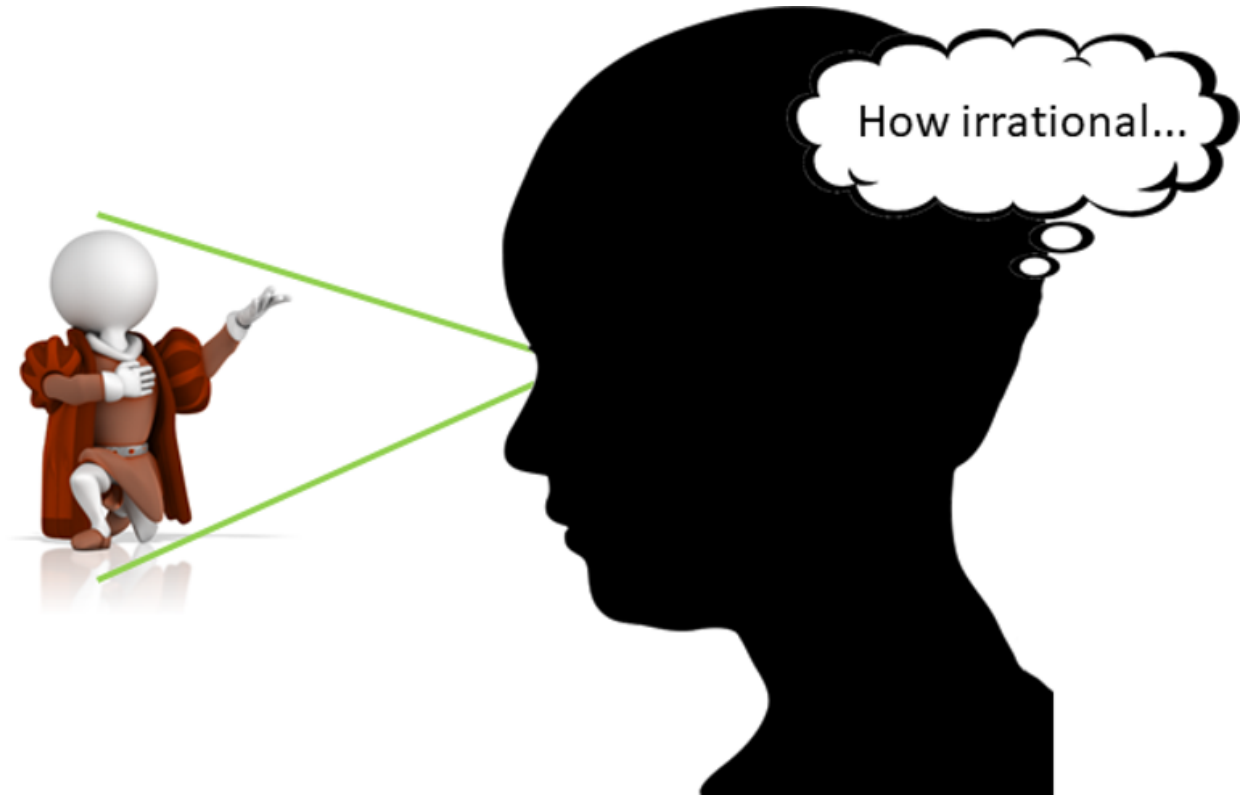
Parent project: this is a subproject of the value learning project.

Background

This idea grew out of a conversation with Matija Franklin and Rebecca Gorman.

My Occam's razor paper with Sören Mindermann demonstrates that human behaviour does not provide enough data to define human preferences and values. I've grounded human values, ultimately, in our mental models, including our assessments about our own irrationality/preferences and the irrationality/preferences of others.

But it is very difficult for an algorithm to use this information. In order for an algorithm to use people's judgements to assess the irrationality or preferences of another person, the algorithm would have to know that some human action was going on, that another human was reacting to it, how they were interpreting it, that they were judging it to be irrational/rational/preference-relevant/etc... and that this judgement was to be taken as normative or as a label. So, not only is the person thinking "how irrational", but that interpretation is to be taken to be correct.

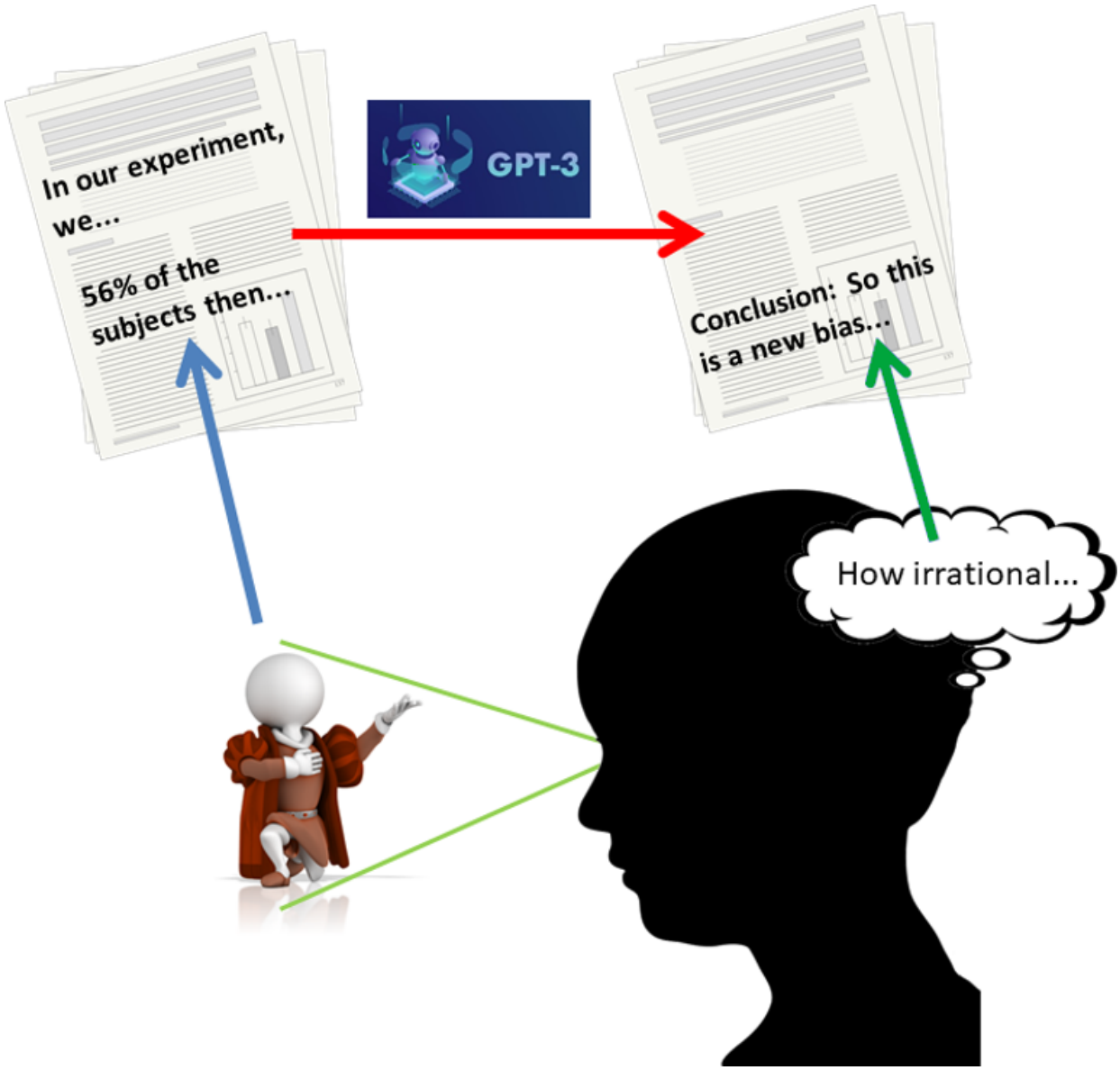


Setup

There is a shortcut to part of this process, though: psychology papers. Let's take the anchoring bias. As I've noted, we could see the anchoring bias as a preference to name numbers close-to-numbers-we-have-recently-heard. But no-one thinks about it that way.

If we look at a seminal paper on anchoring bias, it details how students were presented with products, asked if they would buy those products for the last two digits of their social security numbers, and were then asked to state the actual amount they would pay for them (the experiment). The responses were recorded (the results) and the authors commented on how they demonstrated an (irrational) "anchoring and adjustment" heuristic. This commentary might be in the abstract, introduction, or conclusion sections (or spread across other sections).

What if a fine-tuned GPT-3 or GPT-n was able to generate the author's comments from just the experiment and results? In that case the GPT-3 would be able to generate textual assessments of irrationality or bias, from textual descriptions of experiments or situations:



The blue and green arrows are ungrounded; the algorithm is merely learning the red connection. But that solves a large chunk of the problem. This algorithm would be able to infer textual descriptions of biases and preferences, from textual descriptions of situations.

Grounding the descriptions of biases and preferences (the green arrow) shouldn't be too hard: we just need to point out that words such as "values", "biases", "preferences", "heuristics" and so on, mean "values", "biases", "preferences", "heuristics" and so on.

Grounding the other side (the blue arrow) is more tricky - going from a situation to a textual description of it. This move is especially difficult if we need to add hypothetical situations as well. But it is still essentially empirical, asking how a human would describe a situation, and, as such, it can be trained on human descriptions of situations in general.

So it seems that not only does human text encode a lot of preference-relevant information, but that we can access this information, in part at least, without too much difficulty. I've been mainly thinking in terms of learning biases, but the general idea might allow us to unlock a lot of implicit human knowledge, hiding in our texts.

Research aims

1. See if psychology research papers can be easily separated into "descriptions of experiments or situations" and "human interpretations of the experiments or situations".
2. Generate textual assessments of irrationality or preferences from textual descriptions of situations or experiments.
3. Check the reliability of the textual assessments, and whether they can be used as labels for assessing preferences.
4. See what other types of human judgement information is coded, ungrounded, in texts of various types.

Finding the multiple ground truths of CoinRun and image classification

Crossposted from the AI Alignment Forum. May contain more technical jargon than usual.

Research projects

I'm planning to start two research projects on model splintering/reward generalisation and learning the preferences of irrational agents.

Within those projects, I'm aiming to work on subprojects that are:

1. Posed in terms that are familiar to conventional ML;
2. interesting to solve from the conventional ML perspective;
3. and whose solutions can be extended to the big issues in AI safety.

The point is not just to solve the sub-problems, but to solve them in ways that generalise or point to a general solution.

The aim is to iterate and improve fast on these ideas before implementing them. Because of that, these posts should be considered dynamic and prone to be re-edited, potentially often. Suggestions and modifications of the design are valuable and may get included in the top post.

Generating multiple rewards and objectives

Thanks to Rohin Shah, Ramana Kumar, and Rebecca Gorman.

Parent project: this is a subproject of model-splintering (value extrapolation).

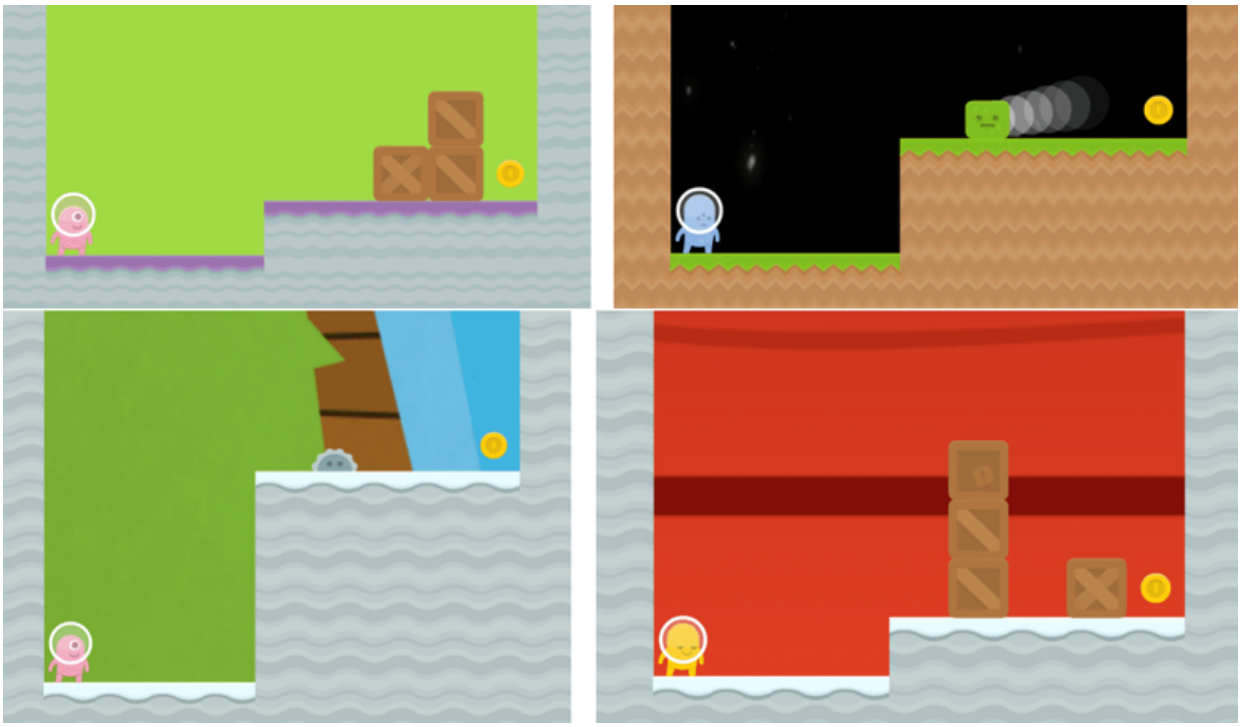
Generating multiple rewards

Suppose that an agent is trained on videos of happy humans. The wireheaded reward is for it to create similar videos for it to watch. We'd prefer that it instead worked to make real humans happy.

But for that to be possible, it needs to consider that "make real humans happy" is even a possible reward function. Thus it needs to generate multiple reward functions that can explain the same data.

Working in CoinRun

CoinRun is a procedurally generated set of environments, a simplified Mario-style platform game. The reward is given by reaching the coin on the right:



Since the coin is always at the right of the level, there are two equally valid simple explanations of the reward: the agent must reach the coin, or the agent must reach the right side of the level.

When agents trained on CoinRun are tested on environments that move the coin to another location, they tend to ignore the coin and go straight to the right side of the level. So that one reward is chosen by default. The aim of the research project is to make the algorithm generate multiple (simple) reward functions that explain the initial data, including the "reach the coin" reward. This needs to be done in a generalisable way.

Multiple image classifications

Consider the image classification task of classifying images of huskies versus images of lions:

A naïve image classifier could be trained on images of this type. The simplest one would probably become a brown-versus-white classifier. We'd want to force the algorithm to generate more classifiers (more "reward functions" for the task of correct classification).

One way to do that is to give the algorithm many other images, unlabelled. Then, in a semi-supervised way, the AI will figure out the key features of these images. Then different classifiers will be trained on the original image data, using these features.

The ultimate aim is for the algorithm to produce, eg, one classifier that classifies through colour, another that classifies through landscape, another through husky versus lion, etc...

Research aims

1. Figure out how to generate multiple policies from the same data.
2. Figure out how to generate multiple simple reward functions or classifiers from the same data (there is a connection with reward modelling).
3. Ensure that these reward functions (or, at worst, policies) are "independent" from each other. This involves figuring out a general definition of independence (in CoinRun, the two independent reward functions - coin and right side of the level - are clear).
4. See if these reward functions "span" the set of reward functions in a useful way - ideally, every possible reward function should be a function of these simple reward functions.
5. Generalise this approach beyond CoinRun, image classification, and similar setups, to general reward data in general situations.

This is a simpler version of the project presented here, generating multiple reward functions from a simpler environment.