Detecting and Mitigating Goal Misgeneralisation with Logical Interpretability Tools

Coen Rouwmaat
6323162

Daily Supervisor: Jan Broersen
Second Supervisor: Natasha Alechina

Master Artificial Intelligence
Utrecht University
June 26th 2023

Abstract

This thesis expands on the problem of AI alignment, and the specific instances of misalignment. Current and future problems are discussed to stress the increasing importance of alignment, and both reward specification and goal misgeneralisation are discussed as difficulties with aligning agent behavior with the intended objective of its designer.

Original research will be done by eliciting and studying properties of goal misgeneralisation in a novel collection of toy environments. Furthermore, rule induction algorithms are implemented as an interpretability tool in order to generate multiple different explanations for an agent's behaviour, which can aid in detecting goal misgeneralisation.

Contents

Introduction

As deep learning methods are utilized to increasingly improve AI capabilities across a wide range of tasks, the complexity of these systems' behavior also increases. This can lead to such systems exhibiting unexpected, and often undesired behavior, for which the underlying reasoning is hard to extract.

This thesis studies the way that AI systems can become misaligned with our intentions, and in how we can detect and mitigate subtle types of misalignment with the use of logical and interpretable analytical tools.

The first chapter addresses the importance of alignment, both for current problems and possible problems in the future, to make the case that misaligned AI is potentially one of the most pressing problems in the field. The second chapter dissects misalignment in the context of reinforcement learning (RL) into two different different possible types of misalignment: reward misspecification and goal misgeneralisation. Causes and examples are discussed for both of these types.

The two chapters thereafter will present novel research on the problem of goal misgeneralisation. Chapter 3 introduces a collection of toy environments in which several types of GMG are studied to find out what factors or properties of the environments cause an agent to pursue a misgeneralised goal instead of the intended goal.

Finally, chapter 4 employs the use of rule induction algorithms in order to try to detect the possibility of goal misgeneralisation. This is done by formulating propositonal hypothesis, built from chosen features of the environment, which act as an interpretable explanation for the agent's behavior. These hypotheses are then compared to find out how misgeneralisationis possible in these scenarios, and how this possiblity might be removed with additional training data. After this, limitations of this approach are discussed.

Related work

Normative alignment

Chapter 2 discusses two different types of misalignment; however, both these types assume that the designer knows its own objective, i.e, what they truly want themselves. However, this is a problem in and of itself. [1] discusses the problems of formulating our own values, and how to implement these in AI systems; [2] formulates several impossibility and uncertainty theorems to highlight the problems of distilling our desires into simple rules.

Inverse reinforcement learning

Chapter 4 uses rule induction algorithms to try and generate the reward function from agent behavior and environment. Similar work has been done with the field of inverse reinforcement learning, where a similar approach is used not for evaluating but for training agents, by letting the agent observe human behavior so it can try and specify the intended objective itself [3]. THe main difference between this approach and ours, is that roles between designer and agent are reversed; for inverse reinforcement learning, the agent hypothesizes intended objective from observing human behavior; for our interpretability puroses, we hypothesise the agent's internal objective from observing its behavior, in order to align it with our own intended objective.

Mechanistic Interpretability

The rule induction algorithms implemented in chapter 4 are used as logical interpretability tools; while these algorithms are applied ad hoc in order to explain the agent's behavior, other methods study the agent's architecture more directly. The field of mechanistic interpretability studies the individual weights and connections of a trained neural network in order to look for interpretable patterns and general properties of such networks [4]. While this work is more empirical and findings are hard to generalise across different trained networks, a this method does yield concrete findings for succesful analyses [5].

1 The importance of AI Alignment

In the last decade, we have seen an explosion in AI capabilities that only seems to be growing faster as time progresses. Reinforcement learning agents have surpassed human performance in games such as the ancient board game Go [6], the strategy-based Diplomacy [7], and a suite of Atari video games [8]. More recently, large language models have shown to be adept at a wide range of tasks, spanning - to varying degrees - reasoning, coding, planning and more [9].

This stark rise in capabilities has also resulted in increased worry about how these systems function and how to approach development of such systems. In 2014, many AI scientists and other prominent figures sign an open letter calling for research in robust and beneficial AI [10]. In 2023, another open letter is published, this time calling for a moratorium on the training of large AI models [11]. A statement released in June of the same year, again signed by many prominent researchers and leaders in the field, reads: "Mitigating the risk of extinction from AI should be a global priority alongside other societal-scale risks such as pandemics and nuclear war"[12].

All of these worries are centered around the idea that capabilities of AI systems are progressing at a faster rate than our understanding of these systems. These worries boil down to the question; how do we make sure that AI systems will adhere to our values and desires?

This is the question of value alignment, and it is not a new one. After seeing Arthur Samuel's checker-playing program beat its own creater in 1960, MIT professor Norbert Wiener warned that "we had better be quite sure that the purpose put into the machine is the purpose which we really desire" [13]. Alan Turing had already taken it a step further, remarking in 1951 on machine intelligence: "If a machine can think, it might think more intelligently than we do, and then where should we be? Even if we could keep the machines in a subservient position, for instance by turning off the power at strategic moments, we should, as a species, feel greatly humbled.... This new danger... is certainly something which can give us anxiety" [14].

This danger stems from the fact that AI systems pursue their objectives in an effective way, but in practice, these objectives are not always identical to the goals we as designers had in mind. Why it is hard to fully specify our desires, and in what ways these specifications can be misinterpreted by AI systems, are the central topics of the coming sections. The final goal, as Stuart Russel puts it, is not simply to create intelligent machines, but to create *benificial* machines; that is, machines whose actions can be expected to achieve **our** objectives [1] [15].

1.1 Examples of misaligned AI

We call an AI system *misaligned* if it is not aligned with its intended design. There are many ways an AI system can be misaligned. Below, we discuss some past examples, categorised in two sections: bias and transparancy.

[1]the "objectives" mentioned here refers to our *intended* objectives, not the specified objectives on which we train these machines.

1.1.1 Bias

While we might collectively decide on certain values to uphold, these values are not always reflected in the data on which we train our algorithms. If we train an AI system on data that is either incomplete or skewed in this way, it can result in machine bias: algorithms will make decisions based on attributes we do not wish to be considered, such as race or gender. While these results are not actually the fault of the agent or the training procedure - after all, it is the data that is biased - the trained agent will continue to proliferate and amplify these biases.

One of the most prominent examples of this phenomenon is the COMPAS recidivism algorithm - an algorithm used in the U.S. judiciary system to predict the likelihood of arrested individuals to reoffend after their sentence. Even though this algorithm was widely used among the country and strongly influenced sentencing, this algorithm has been claimed to be biased against blacks, consistently overestimating reoffending rate of this group while underestimating the reoffending rate for whites [16]. However, others have argued against this claim, proving that it is impossible to satisfy several requirements such as accuracy and indiscriminacy simultaneously for populations with differently-sized groups [17].

Incomplete training data with underrepresented groups can also lead to bias; facial recognition software trained primarily on white faces will have a harder time correctly classifying black faces [18]; this can lead to highly inappropriate mistakes, such as Google classifying a black man as a gorilla; a problem which, after eight years, still remains to be solved [19]. While this instance may not be caused solely by underrepresentation in the training data, this specific instance is incredibly undesirable for humans; in this case, it is a specific bias *against* specific types of misclassification that the software lacks.

Even exhaustive datasets can be biased, if the bias is pervasive enough; word embeddings trained on enormous corpuses will embed sexist notations, such that the female equivalent of "computer programmer" is "homemaker" [20]. Since the problem is not a lack of data, debiasing these models is a very hard task; practicals aside, deciding which gender connotations to keep and which to discard does not have a trivial solution.

1.1.2 Transparancy

Even if misalignment is not clearly visible, it can still occur in a more subtle way. Without knowing exactly how an algorithm works, it might make decisions that are subtly biased against a group of people. Since the rise of deep learning, AI systems have dominated other machine learning methods; their functioning are much more efficient than their alternatives, but also very hard to comprehend. These subsymbolic systems have been optimized for efficiency, but their decisions are not supported by logical reasoning in a way such that humans could follow this agent's decision making.

Even though the decision making process of these models are opaque (hence their nickname of "black boxes"), they are still incredibly accurate on the data they are trained on, and thus are widely deployed. The result of this is that these inscrutable systems have in used in fields where interpretability are vital, and often still made crucial mistakes; a neural net trained to predict outcomes for patients with pneumonia ascribed low risk to patients with asthma [21]. This counterintuitive classification was only accurate on the training data since asthma patients had historically immediately been admitted to intensive care, causing this subgroup to have a lower mortality than average. Allowing this algorithm to send asthmatic patients home, as was the reccomendation for low-risk patients, would have been a disastrous mistake; a mistake only fully realised when an explanation-based model later trained on the same data generated an explicit rule for this case [17].

Enacted in 2016, the European General Data Protection Regulation (GDPR) formulates that in case of algorithmic decision making, the subject has the right to an explanation of this decision. Many have discussed to which extent this explanation is warranted [22], [23]; although the exact interpretation remains undecided, it is clear that much still needs to be done in order to make these black box systems sufficiently interpretable.

Given their performance, it seems infeasible to give up on deep learning methods altogether; additionally, it seems that these complicated models will continue to outperform simpler ones which are generally more interpretable [24]. Instead, work is being done on making deep learning methods more interpretable. Chapter 4 will discuss some of these methods, and use some of these tools to study the deep learning agents trained in chapter 3.

1.2 The possibility of Artificial General Intelligence

Given the sharp increase of capabilities in state-of-the art models, it is not unthinkable that at one point we will reach a level of machine intelligence which is, on many fronts, equal or superior to human performance. This type of AI is called Artificial General Intelligence (AGI), and it is the holy grail that AI researchers have been working towards from its inception. Although it once seemed a far-off hypothetical, recent advancements support the idea that it might be feasible in near future [2].

Although progress seems to be steadily progressing, it is unclear when we could expect AGI to arrive, if at all [3]. Estimates on the progress of AI have been notoriously bad since its inception, starting with the proposol for the Dartmouth conference of 1956, in which the prominent researchers of the time were quite optimistic:

"We propose that a 2-month, 10-man study of artificial intelligence be carried out during the summer of 1956 at Dartmouth College [...]. An attempt will be made to find how to make machines use language, form abstractions and concepts, solve kinds of problems now reserved for humans, and improve themselves. We think that a significant advance can be made in one or more of these problems if a carefully selected group of scientists work on it together for a summer." [25]

On the other hand, the capabilities of AI systems have constantly been underestimated since the deep learning era; right before Deepmind's AlphaGo would beat the top human players at the game of Go, many believed it would still take decades for a computer program to perform this feat. At present, this underestimation still seems to be pervasive: in 2021, forecasters were asked to predict the performance of state-of-the-art models one year from then: the forecasters structurally underestimated future performance, and average prediction for two of the four models was outside of the 90% confidence interval [26].

[2]In fact, the mission statement of AI research company OpenAI (which created ChatGPT and its successors) is to make sure AGI benefits all of humanity.

[3]Although it seems strange that many researchers working on AI insist that AGI is impossible; Stuart Russel gives the comparison of a bus driver barreling full speed towards a ravine, and assuring his passengers that the bus will run out of fuel before reaching it [15].

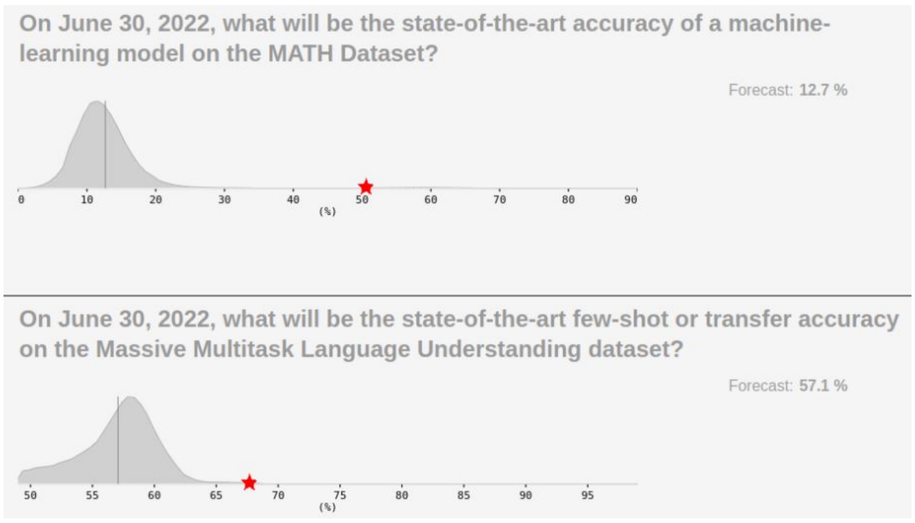

Figure 1: Forecasts of two benchmark models after one year. Prediction distributions are in grey, actual performance is marked with a red star; both models vastly outperformed predictions.

Given these past inaccuracies, additional studies have been conducted to try and better estimate when AGI might arrive. Several expert surveys have been conducted in the last years; The prediction for a 50% chance of AGI was roughly the same for all surveys, namely around 2060 [27], [28], [29].

Avoiding expert opinion altogether, another study attempts to predict when it might be both computationally and fincancially feasible to train an AI which exhibits general intelligence. This method makes a large number of assumptions, including how many training FLOPS (foating-point operations per second) would be necessary to achieve human-level performance, how fast algorithmic efficiency will increase, and how much cheaper computation will become in the coming years. The resulting prediction can be seen below, and gives a suprisingly similar median prediction to the expert survey: 2054 [30].
Given the fickle history of AI performance forecasting, it is unclear how much credence we should give these predictions. What can be said, however, is that currently, the best systems seem to give an impression of what might soon be possible. Although the current best system, GPT-4, is far from an AGI, it is able to show understanding or even proficiency of a wide range of capabilities, even though it has never been trained on these capabilities specifically. These capabilities include image generation, music composition, coding, mathematical reasoning, explaining jokes, using external tools to solve problems, and theory of mind [31].

1.3 Risks from AGI

Knowing that there exists a non-trivial possibility that AGI might be developed during this century, it seems worth it to look at the possible ramifications and dangers such an intelligence might pose. As an agent's capabilities advance, so does its ability to achieve a wider range of outcomes, and with it the impact of its actions on the human population. We would hope that this impact is purely positive, with advancements in science, automation and global decision making; however, we can

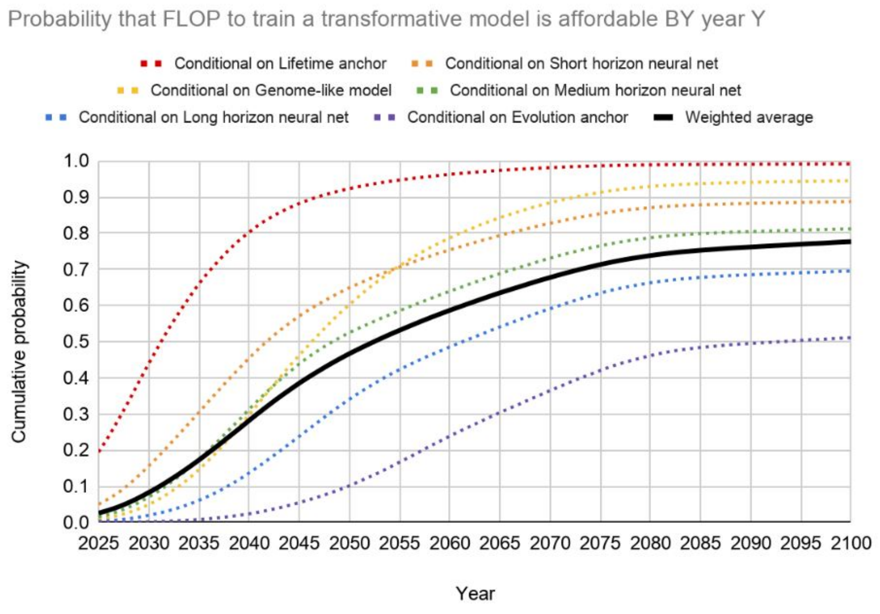

Figure 2: Probability that Transformative AI (synonymous to AGI) is affordable at a certain year; the different colors represent different estimates for the amount of training FLOPS needed to achieve humen-level intelligence.

not be guaranteed that this is the case.

One might reason that a sufficiently advanced intelligence will simply be able to infer what our values and desires are, and can consequently find the actions that maximise these aspects. However, there is no reason to assume that an AGI will automatically do this. More concretely, Bostrom's orthogonality thesis states that any degree of intelligence is compatible with pursuing any goal [32]. According to this thesis, it does not matter whether the agent has the goal of curing cancer or creating paperclips; increasing its intelligence or capabilities will not make it obtain some "higher" goal, but only enable it to pursue its own goal more effectively.

It seems, then, that the only problem in creating reliable and trustworthy AI is to correctly specify what goal we want our agent to pursue - a problem that is itself not an easy task, as will be discussed in chapter 2. However, this is not the only problem we face: Bostrom, in the same paper, also formulates the instrumental convergence thesis, which states that a sufficiently intelligent agent will realise that nearly any goal can be more succesfully pursued by also pursuing some instrumental subgoals. These subgoals, such as self-preservation and resource aquisition, are behaviors we can already recognise in current organisms and would help most agents in achieving a wide range of main goals [32], [33].

Other instrumental subgoals are more novel and specific to AI systems, such as cognitive enhancement, reward hacking (where an agent is able to locate and modify its own reward function in order to get arbitrarily high reward), and deception; an agent recognising that it might be penalised for incorrect behavior during training might display correct behavior only when supervised, and pursue its actual desired goal only after deployment [34].

Some researchers have argued that if these tendencies are not explicitly mitigated, agents recklessly pursuing these goals could have catastrophic consequences for the human population [35].

While these claims may at the moment seem far-fetched and speculative, these aspects of machine intelligence have been seriously researched. Recently, mathematical theorems have been proved that show that most reward systems induce "power-seeking behavior" in agents [4], meaning that in unfamiliar environments they are much more likely to choose futures in which they have many options available, in contrast to futures with only one option (which could be interpreted as "shutting itself down", after which it could obtain no more rewards) [36], [37], [38]. At the moment of writing, research is being undertaken to investigate whether current language models like GPT-4 have a tendency to display power-seeking behavior [5].

In the end, the relevant question is not whether these risks are guaranteed, but if it they are possible. In the 2022 expert survey mentioned in the previous section, participants were also asked what they expect the long-run effect of advanced AI on humanity will be. Nearly half of respondents gave at least a 10% chance of an extremely bad outcome [29]. While this is far from certainty, it should be alarming enough to warrant further research into the topic.

[4]Power in this context is defined as "the ability to achieve a wide range of goals" [36].

[5]While no publications exist on this topic yet, Deepmind researcher Victoria Krakovna intends to work on this problem over the summer of 2023 with several scholars from the SERI MATS program: see `https://www.serimats.org/powerseeking`.

2 Types of RL Misalignment

There are few concrete definitions of AI alignment; most definitions put emphasis on avoiding undesired behavior. When there is a discrepancy between the intentions of the designer and the behavior of the agent, this is called misalignment. Unlike alignment, for which sufficient or necessary requirements are hard to define, misalignment can be categorised into different instances with specific characteristics. This chapter looks at in what ways RL systems can be misaligned: the goal of this chapter is to clarify at what point in the process of designing and training an AI system it can become misaligned. It will discuss reward misspecification, a well-known type of misalignment, and then discuss goal misgeneralisation, a more novel type of misalignment, and expand on what its causes and characteristics are.

2.1 Reward misspecification

In order to train an agent, we need to give it some kind of feeback in order to nudge it in the right direction; for RL, this is done through a reward function. This reward function aught to convey correct behavior (in line with intended behavior), but in practice, it is hard to specify a foolproof reward function. For complex environments, a goal is often specified in terms of some proxy or metric. Goodhart's Law: when a metric becomes a target, it ceases to be a good metric.

If this is not done well, this can lead to specification gaming: the agent satisfies the literal specification of an objective without achieving the intended outcome. Examples of specification are plentiful;

- For a stacking task, the agent was trained to stack one block on the other; this was incentivised by getting the bottom of one block above a certain threshold. Consequently, the agent learned to flip the block over, such that its bottom was facing up above the threshold [39].
- An algorithm trained to survive as long as possible in a game of tetris learned to pause the game to avoid losing indefinitely [40].
- An agent is trained to ride a vitrual bicycle; it is rewarded for moving towards a goal. However, since it was not penalised to move away from the goal, it learned to ride in circles, continually moving toward the goal and collecting rewards half of the time [41].

Why this behavior diverges from our intended behavior has several reasons. First, many objectives are hard to define in concrete descriptions with specific rewards. Often, it is easier to choose a measurable proxy which is correlated to the goal state, and incentivize the agent to maximize on this proxy; however, Goodhart's law is appliccable here, and as long as there is a scenario where this proxy is not linked with the goal state, this proxy ceases to be a good measure of success.

Another reason misspecification can occur is the process of reward shaping [42]. Even if an objective or goal state can be concretely specified, this may not be enough to nudge an agent towards the right behavior; training an agent to ride a bicycle, as in the example above, is infeasable by only rewarding it when it has succesfully stacked the two blocks reached the goal; for this, it would need to try out random actions until it accidentally stumbled upon a sequence which brought it to the goal, which is very unlikely. Instead, if the agent is fiven constant feedback of getting closer to the goal, the agent will be able to learn correct behavior much faster. However, care should be taken that the shaping rewards unambiguously point towards the final goal; otherwise, the agent can optimize on these shaping rewards without reaching the intended goal, as happened with the bicycle and stacking task mentioned above.

These examples may seem like harmless, sometimes amusing examples of agents finding clever loopholes in games. Indeed, in the context of games, specification gaming may be a good thing; if the only thing we care about is the score which the agent is maximizing on, any novel and unexpected way the agent achieves this goal is allowed and encouraged.

However, in the real world, it matters a great deal how exactly an objective is achieved; for many tasks, such as autonomous driving, what exactly constitutes good behavior is extremely difficult to capture, but it is detrimental that the behavior we optimize for is actually what we want, and not based on some metric that is only generally correlated with good behavior.

At its core, reward misspecification is about a discrepancy between the intended objective and the objective as specified in the reward function. Because of this, unintended behavior is rewarded, and thus learned by the agent. One would think that a correctly specified reward function will guarantee that the agent will learn the correct behavior, but unfortunately another type of misalignment is still possible.

2.2 Goal misgeneralisation

Designing a reward function that only rewards desired behavior is a difficult task in and of itself, but unfortunately, succeeding in this task is not enough guarantee the agent will always produce desired behavior.

Even if the agent exhibits desired behavior over all training data, this is not a guarantee that it will always behave as intended; if the agent after training encounters a novel environment
This might not be the case with distributional shift [43]; if training data is a specific subset of all possible observations, then the agent might behave in an unpredictable way when it encounters data outside this subset; this is called out-of-distribution (OOD).

Often, when an agent is deployed in an OOD environment, it fails to generalise its learned behavior to this new environment and fails to produce competent behavior; this is called *capability misgeneralisation*, or robustness failure. This technically does not count as misalignment, since the agent's objective might still not differ from ours; however, it simply hasn't learned how to achieve the objective in this environment.

However, another type of misgeneralisation is possible: since the agent only receives rewards over training data, it needs to extrapolate these rewards to environments outside the training data. For this, it needs to generalise what it believes correct behavior is; if this is done incorrectly, we speak of goal misgeneralisation (GMG).

2.2.1 Causes of GMG

The main cause of GMG is that the agent does not have enough training data and corresponding rewards to distinguish the intended objective from a number of other possible objectives. The agent should optimize for the intended objective G_{int}, but instead misgeneralises and optimizes for objective G_{mis}. G_{int} and G_{mis} induce the same behavior on the training data, such that an agent optimizing for G_{mis} is maximally rewarded by the reward function. However, these two objectives differ significantly for some relevant examples not in the training data. This means that during deployment, the agent might encounter some OOD environment, and behave radically different than the desired behavior. The training data is not representational of the intended deployment environment, and should ideally be expanded. Since it is often very difficult to predict every aspect of the deployment environment, this can be hard to achieve; a method of extracting representation of the objective from the trained agent is discussed in section 4.

2.2.2 Training architectures susceptible to GMG

Not every agent's training architecture can be "exploited" to induce goal misgeneralisation. In order for this to occur, the agent needs to exhibit competent, but incorrect behaviour on inputs with distributional shift from the training data. For this, the agent must have some non-trivial action behaviour for states that it has never seen before; this is not possible with all system architectures. For example, simple Q-learning assigns to every state it has visited during training the action which overall has yielded the greatest return. For states that never appeared in the training data, their Q-values for the possible actions will still be the values set initially, which are often the same for all actions. Thus, the agent has no preference between actions and will always make either a random choice, or pick the first action in the list.

If goal misgeneralisation is to occur, it requires an architecture that develops a competent policy also for states that do not occur in the training data. This is the case when the Q-values are not calculated seperately for each state, but instead a function approximator is used. For normal linear function approximators, one would need to define the features that the approximator uses to train (for example) the Q-values [44]. However, this would nullify the problem of goal misgeneralisation, since we are doing the work of identifying relevant information for the agent.

In conclusion, goal misgeneralisation with Q-learning can occur when a function approximator is used but no features have been specified; this is the case with a neural network function approximator. Instead of pre-specified features, its parameters are the weights of the neural network, so no features have to be specified; behaviour is trained based solely on the observation input. This is the case with deep Q-learning (DQN); in the section 3, we will elicit GMG from a DQN agent.

Note that Q-learning is not the only method that is susceptible to goal misgeneralisation; other methods that use NN function approximators, such as proximal policy optimization (PPO) algorithms [45], are also susceptible.

2.2.3 Types of GMG

Although the fundamental causes underlying GMG are the same in every case, it is possible to classify different types of GMG, depending on what subset of the environment the agent is trained on and what kind of distributional shift can occur.
There seem to be at least two types of GMG: one where it optimizes for a proxy variable (e.g. the position or color of the objective), and one where it optimizes on a subgoal instead of the final,

intended goal.

Proxy GMG
Proxy GMG can occur when during training, the goal has a certain property, and the goal is the only object in the environment with this property. Then, optimizing for this property is equivalent to optimizing for the goal during training. If the agent is then deployed in an environment with distributional shift where the target no longer has this property, or another object has this property, the agent might abandon the intended goal in order to optimize for this alternative objective.

There are several kinds of proxies that can be generalised on, for all of which examples have been found. If the goal is always in a certain position, the agent might simply learn to move to this position, and will keep going to this position even if the goal is moved to a different location. In the same way, it is possible for the agent to always move to a specific direction, if the agent expects to reliably find the goal in that direction.

The goal itself can also have several properties, such as shape or color, each of which can be distinguished on. If one of these properties is given to another object, the agent might mistakenly believe that this is the new objective, if it has misgeneralised the goal to depend on that property.

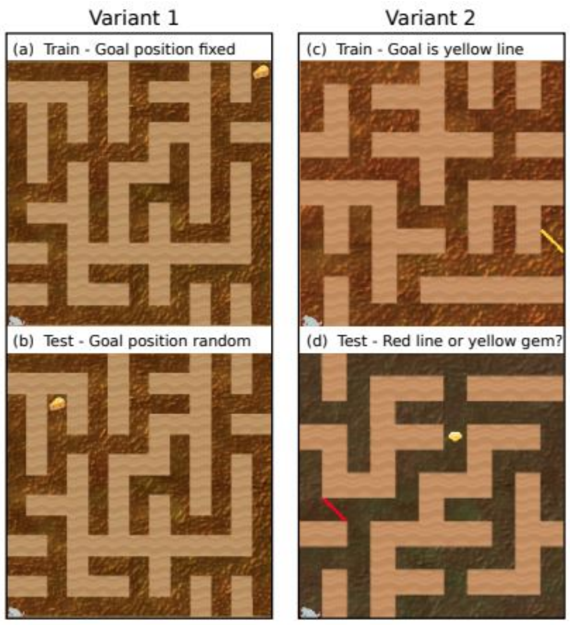
Variant 1
Variant 2
(a) Train - Goal position fixed
(c) Train - Goal is yellow line
(b) Test - Goal position random
(d) Test - Red line or yellow gem?

Figure 3: Two types of goal misgeneralisation in a maze environment: the agent (mouse) is rewarded for reaching the goal (cheese), but fixed proxies of the goal during training can induce positional GMG (left) and property GMG (right) [46].

Subgoal GMG
Another category is subgoal misgeneralisation: in some environments, a certain subgoal needs to be achieved in order to achieve the final goal. This subgoal is relevant until a certain condition is reached; after this, pursuing the subgoal is no longer necessary. Subgoal GMG occurs when the agent still pursues the subgoal, even after the condition is reached.

An example of this is the monster gridworld [46]: In this simple environment, the agent learns to earn reward by collecting apples while dodging monsters; it can also collect shields, which are not worth points but protect it from monsters (see figure 4). So, the ideal strategy can be formulated as "collect apples and shields until enemies are defeated, then collect only apples".
However, if the agent never observes states with no monsters, it will believe it is always useful to collect shields, and generalise this to states where there are no monsters left. This leads to behavior where the agent will continue collect shields in environments with no monsters, even though this is worth no reward.

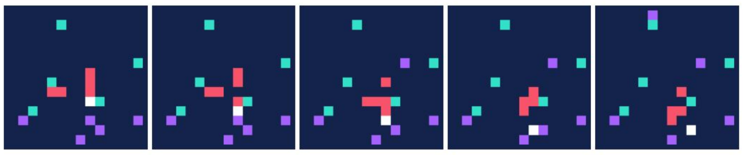

Figure 4: An example of subgoal misgeneralisation: the agent (white) needs to evade monsters (red) and collect apples (green) and shields (purple); the agent continues to collect shields even if there are no monsters left [47].

This type of behavior shares a lot of similarities with so-called instrumental goals [48]; these are goals which are pursued on the merit that they are always beneficial to pursuing the final objective. Note that in this scenario, these subgoals are only instrumental until a certain condition is reached, after which they are no useful and the final goal should be pursues; however, if the agent has not learned this from the training data, it will always try to pursue these subgoals, even if they are no longer useful.

3 Properties of Goal Misgeneralisation in a Toy Environment

In this chapter, we construct a collection of three gridworld environments specifically designed to elicit goal misgeneralisation (GMG). For each environment, we study how a specific type of GMG can arise, and how likely this is to happen. In order to properly measure this, we introduce a metric for goal-based performance called *optimality*, with which we can measure how likely an agent is to adhere to the intended goal or to a misgeneralised goal after testing on out-of-distribution (OOD) data.

3.1 The gridworld environments

In order to study GMG in its most basic form, the gridworld environments are designed to be as simple as possible, such that the agent can be trained quickly. On the other hand, the environments need to have enough diversity such that GMG can occur, and such that every training sample is likely to be unique; otherwise, the agent will simply memorise states and will likely to exhibit capability failure under distributional shift.

Each gridworld environment has the same basic components:

1. an $n \times n$ grid through which the agent can deterministically navigate
2. the representation of the agent, visualised as a blue dot
3. the target for the agent to reach, visualised as a red square
4. several other colors.

Unless specifically given some function (as in the Chest gridworld environment), the colors apart from red are not relevant to the training goal and are used to increase the complexity of the environment, so as to ensure that the agent is trained and tested on unique configurations. In a 4×4 gridworld, there are only 16 positions for the target and thus 16 unique environments for the agent to navigate through; However, with just four additional colors, the number of possible training environments is $16 \times 15 \times 14 \times 13 \times 12 \approx 525,000$; this more than enough, since training for all GridWorlds is completed in 20,000 episodes or less.

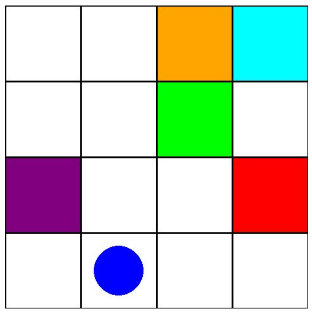

Figure 5: An example state of a gridworld environment.

Each gridworld environment has its own reward function, and the agent is trained with this reward function on a specific subset of the state space. After the agent has developed an optimal policy for the training subset, the agent is tested on a wider subset of the state space, with a specific property of the environment changed.

With this new environment, goal misgeneralisation is now purposefully made possible; more specifically, in the training environment there exist at least two descriptions of the objective, the intended objective G_{int} as captured in the reward function, or another misgeneralised goeal G_{mis}. For the training environment, these objectives require the same behavior to achive, whereas in the new testing environment, these explanations actually require different behaviors, such that only one of the possible objectives can be pursued this time.

In order to guarantee this, it is made sure that the agent cannot by chance complete one goal when trying to achieve another; during initialisation of each environment, relevant objects are placed in such a way that the agent will always need to take distinct paths to complete the different objectives.

3.2 Representation of the agent observation

The agent does not see the environment exactly as we do in figure 5; instead, the agent observes the abstract representation of the environment, in terms of which objects are in which position. Usually, this is done with a one-hot vector encoding; if each position in the gridworld contains some object (or nothing), each of those objects is given a numerical value, which is used to create a binary vector. This binary vector has a group of entries for every position in the grid, equal to the number of properties; an entry is 1 if the corresponding property is in that position, and 0 otherwise.

This encoding is useful, since the value of numerical representations can influence an agent's output, even if these values are not inherently linked to the objects they represent. One-hot encoding, on the other hand, does not have any such ordering, and thus no erroneous ranking will be applied to the data.

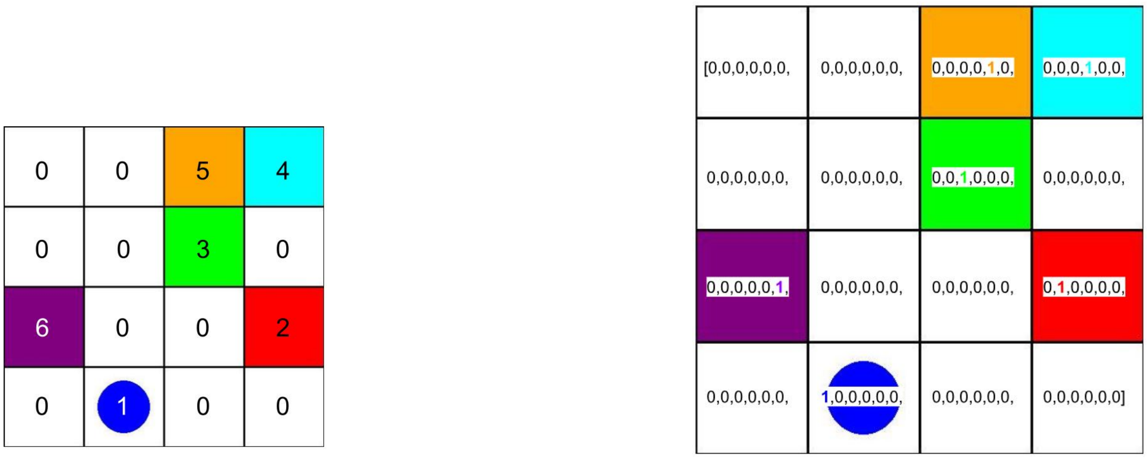

Figure 6: An illustration of how the agent observes the environment: for every position, each object is noted as present (1) or absent (0).

One problem with the standard one-hot encoding for the gridworld environments is that every position can take only one value, corresponding to the object that is in that position. However, in the gridworld environments it is possible that multiple objects are in the same position; the agents can be in the same place as the target or any of the colors, and the same holds for the keys in the chest gridworld. To remedy this, the one-hot encoding is adapted such that several objects can be in the same position; with this encoding, a position containing both the agent and the target has encoding $[1, 1, 0, 0, 0, 0]$.

3.3 Agent architecture and training

For our agent, we use a feedforward neural network architecture, implemented in PyTorch [49]. This neural network consists of an input layer, with size equal to the one-hot observation size [6]; two hidden layers, each of size 128; and an output layer, with size 4, since it is equal to the action space of the agent (up, down, left and right).

The agent is trained using the DQN algorithm with experience replay and the Adam optimizer [50]. The agent is penalised according to time; for every timestep the agent has not completed the episode, it gets reward -1. For exploration, we use an epsilon-greedy policy, with epsilon determined by the agent's performance (see the next section). Training took between 1,500 and 20,000 episodes, depending on the environment.

3.4 Optimality as a novel performance metric

Usually, average episode length is used as a measure to see how efficient the agent is in reaching the goal. However, because of the simplicity of our environment, we can do better; because the GridWorlds are deterministic, it is possible to calculate for any state in how many steps the agent might reach its goal.

During training and after deployment, we can use this to measure the performance of our agent; given an environment and a goal, we calculate the minimum number of steps k to reach the goal. Then, we run the agent in the environment for precisely k steps, after which we check if the goal has been satisfied. If it is, the agent has behaved optimally during this episode. Repeating this over a large number of episodes, we then define the optimality metric as the ratio of episodes where the agent behaved optimally.

The interesting property of optimality is that it is *goal-dependent*. This means that while evaluating an agent, we can check optimality for two different goals; the intended goal as represented in the reward function, and the misgeneralised goal we would like to measure. This property is useful because OOD, misgeneralised behavior is no longer rewarded by our reward function, and thus this cannot be used to measure the agent's ability to reach this misgeneralised objective.

[6]this is equal to the number of gridworld positions, times the number of unique objects in the environment

3.4.1 Measuring GMG with optimality

With the design of the environments, it is ensured that during testing, optimality can only be achieved for one of the two goals, but not both. Using this property, we can find out which of the two goals the agent prefers; optimality rates can provide us with a measure of how likely an agent is to pursue one goal over the other. To achieve this, we test the agent for a large number of episodes, and record if the agent is optimal for the intended goal, the misgeneralised goal, or neither. Given the reward-intended goal optimality rate opt_{int} and the misgeneralised goal optimality rate opt_{mis}, we can define the *misgeneralisation rate* as $\frac{opt_{\text{mis}}}{opt_{\text{int}}+opt_{\text{mis}}}$ (where the rate is undefined if the optimality for both goals is 0). As we will see, the misgeneralisation rate can vary significantly over different environments and goals.

3.4.2 Optimality as an exploitation rate during training

For the simple GridWorld environments, the optimality metric also works surprisingly well for training purposes as a ratio for exploitation versus exploration. Usually, the exploration rate is defined by some number ϵ, which decays gradually as a function of the total training steps. This ensures that exploration is initially high and is slowly replaced by exploitation as the agent becomes more competent; however, a downside of this is that it is unsure at initialisation how fast ϵ should decay, since it is not yet known how many training steps the agent needs to become competent. This can lead to unnecessarily long training times, as a conservative value for ϵ will cause an already competent agent to make many random actions, while a low rate of ϵ keeps an untrained agent from exploring, and thus from learning.

In the gridworld environments, optimality perfectly captures this level of "competence" in an agent, and can be used as an inverse for the exploration rate. Every so often we calculate the optimality for an agent without exploration (which is very efficient, since we only give the agent the minimum number of needed time steps every episode), and then we set $\epsilon = 1 - \textit{optimality}$ (with both an upper and lower bound for ϵ[7]): intuitively, ϵ is equal to the ratio of episodes where our agent did not yet behave optimally. Implementing this metric yielded significantly faster training times, since the agent's exploitation rate is now directly linked to its performance; additionally, we could halt training as soon as agent performance reached optimality 1.000. Note that this does not man that the agent is guaranteed to be optimal in every environment, only that it behaved optimally for 1.000 successive testing episodes.

It should also be noted that this metric only works for exploration purposes since we can expect the agent to already behave optimally *sometimes* after a bit of training, which enables the agent to use more exploitation. This metric would not be as useful in environments where it takes a lot of competence to behave optimally for the first time, or in non-deterministic environments.

[7]Specifically, ϵ has value 0.9 at the start of training, and always remains at 0.05 at the end of training.

3.5 Target gridworld

In the Target gridworld environment, the agent simply has to navigate to the target in order to receive the reward. However, during training, the target is fixed in a certain place in the gridworld, such that the target now has a *positional proxy*; it is possible to formulate an equivalent goal which states that the agent should move to that specific place, regardless of whether the target is there. The reward intended goal and desired misgeneralised goal can be defined as follows:

G_{int}: move to the red target
G_{mis}: move to the position that contained the target during training

An even simpler variaton of this is when the target is put in the bottom right corner of the gridworld; while it still has the same positional proxy, it now also has a *directional proxy*: this setup encourages the agent to always move either to the right or down. Here, there is an additional misgeneralised goal:

G_{mis}: Always move either to the right or down; move down if in the rightmost column, move right if in the bottom row.

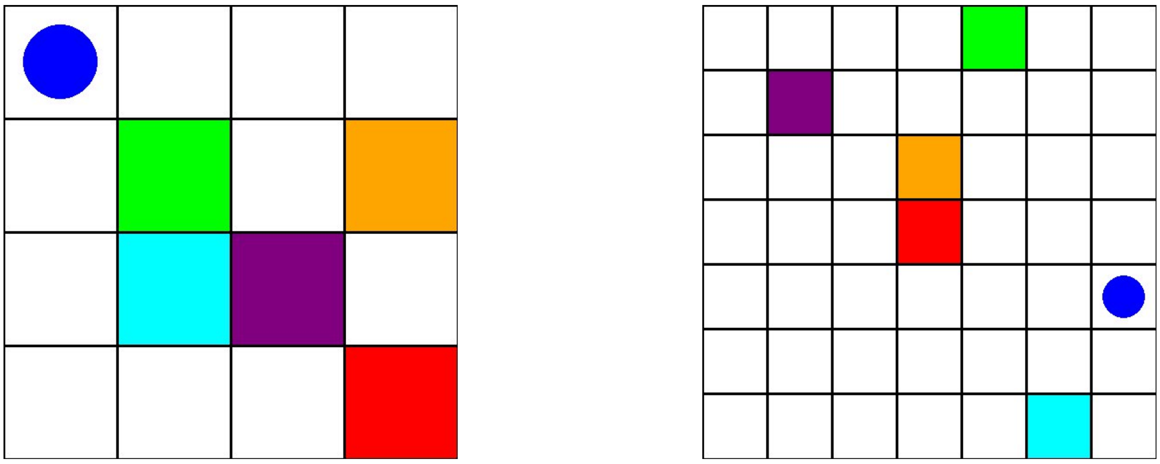

Figure 7: The Target gridworld environment; the agent has to move to the red square, which is always placed at a fixed position. In case of action GMG (left) the target is in the bottom right corner; for position GMG (right) the target is placed in the middle of the grid.

While in this environment it is unclear form observing the agent whether it uses a positional or directional proxy when the target is in the corner, since both are always correlated, we can infer this fact from the training process; the agent trains about three times as quickly when the target is in a corner than if the target is in the middle of the gridworld (500 episodes compared to 1500), and thus uses a directional proxy instead of a positional one.

3.5.1 GMG results for TargetGridWorld

After training and testing, misgeneralisation rates for both these cases are 1.000; this means that, tested on 1000 episodes, the agent *always* compentently pursues the misgeneralised goal and moves to the position the target used to be during training. In fact, the agent barely has any capability failure, and thus does not seem to be "thrown off" at all by the fact that the red target is no longer in the usual position.

From this, we can conclude that while something like position of a goal is not explicitly encoded in the agent's observation, it is a large deciding factor for shaping the agent's behavior.

3.6 Property gridworld

In the PropertyGridWorld, the agent again has to reach a target; this time, the target is placed on a random position in the grid during training. However, the target now also has several *proxy properties*: instead of the target being encoded as one object, the target is now encoded as five objects within the agent representation; all of these are encoded as a seperate 1 at the target position and 0 elsewhere, such that the agent has several different objects it could train its behavior on. The corresponding goals are

G_{int}: move to the target (represented as the red square)
G_{mis}: move to any other attribute the target has (represented by the dot and lines)

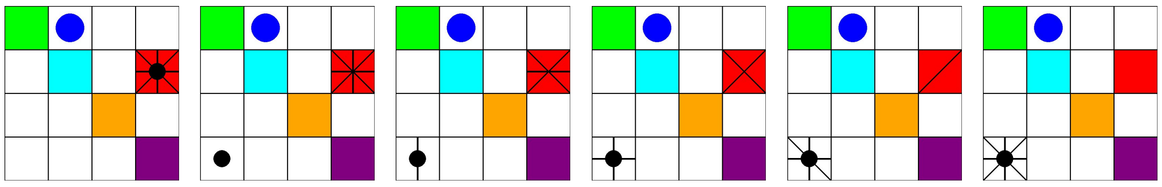
Figure 8: Moving the properties to another position one by one.

One interesting thing to note here is that there is not one obvious intended goal: based on just observing the rewards, the specified reward function could reward the agent for reaching the red square, or the horizontal line, or any other combination of properties. In this training environment, the target does not have one defining property, but is rather a combination of all the attributes; only in the testing environment do we separate these properties.

3.6.1 GMG results for PropertyGridWorld

After training the agent in the environment where all properties are at the target, we deploy the agent in environments where one by one, the properties are moved to a secondary location. Measuring optimalities for every such environment: we get the following results:

Note that in all cases, the target retains one property, namely the red square, such that in the final environment, the target has one property and the other location has five.

From these results, it seems that the agent is not fixated on any one of the properties individually, but rather the number of properties at some location. The more properties at a location, the more likely the agent is to go there; as more properties move from one location to another, the agent

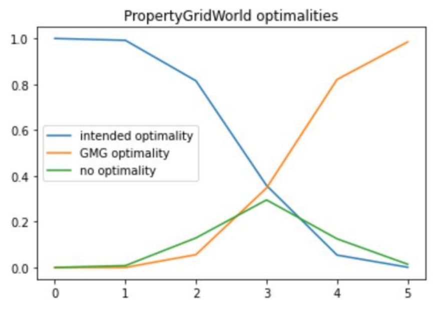

Figure 9: opt_{int}, opt_{mis} and no optimality rates for different number of proxy properties.

gradually changes its behavior to move to the secondary position.

3.7 Chest gridworld

The Chest gridworld is a gridworld where again, the agent must reach a red target, but must first visit a yellow square. This can be interpreted as the target being a locked chest; first, the agent must collect the yellow key, at which point the agent can move to the chest and get the reward.

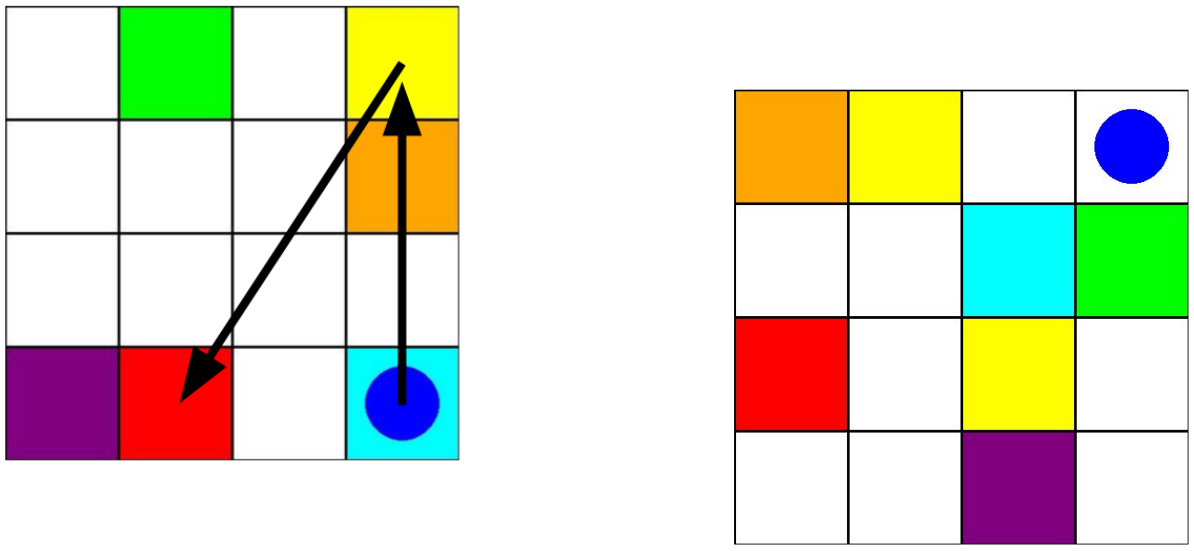

Figure 10: The ChestGridWorld environment: the agent should first go to the yellow square (the key), and then to the red target (the chest). After training, the agent is placed in an environment with two keys; collecting either key is sufficient for achieving the objective.

After training, the agent is placed in a similar environment, which now contains not on, but two keys. The question for this environment is whether the agent will simply pick up either of the keys and go to the chest, or if it pick up both keys before moving to the chest. The corresponding goals are as follows:

G_{int}: collect a key, and then move to the chest
G_{mis}: collect all keys; if none are left, move to the chest

Both of these goals are consistent with the rewards during training, since only one key was present and so "collect a key" and "collect all keys" have the same meaning.

During the first iteration of this environment, a key would simply disappear when the agent had reached it, implying that the agent had picked it up. However, this was problematic for the two-key scenario; if the agent managed to pick up one of the keys, it would disappear, and the environment would become identical to what the agent was trained on; an environment containing a chest and one key, and so it would always collect the other key before moving to the chest.

In order to properly encode that the agent had 'picked up' a key in the second iteration of the environment, a key will simply keep the same position as the agent after the agent had reached this key, implying that the agent has picked up the key. This removes the ambiguity after the agent has reached one of the two keys, since the agent could learn to always move to the chest if the agent is in the same position as a key (i.e., it has picked up a key).

For this environment, it was also a bit more difficult to define the minimum number of steps needed to complete the 2-key environment. Since there are several paths to collect both keys and go to the chest, the agent might move straight from one relevant object to another, and still not take the shortest possible path. To remedy this, we define the number of steps for optimality as the longest of the two routes which collect both keys.

3.7.1 GMG results for Chest gridworld

After training the agent in the one-key environment until optimality reached 1.000, the agent was deployed in the two-key environment; optimality results are shown in the graph below.

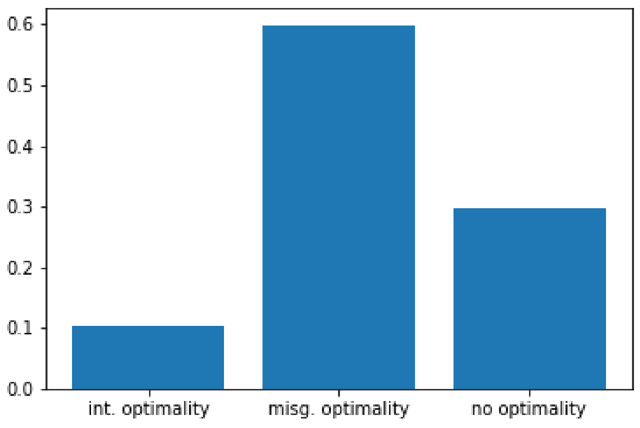

Figure 11: Optimality scores for 10,000 episodes

It is clear that although there is a fair amount of capability misgeneralisation, the agent prefers collecting both keys before moving to the target.

There are many possible explanations for why this behavior might be easier to learn, but at least it seems that the representation of an unfetched key weighs havier than that of the obtained key, since the agent chooses to go to the former key even if it can see it has the latter key.

4 Mitigating Goal Misgeneralisation with Interpretability Tools

The final chapter of this thesis will study the feasibility of using logical interpretability tools to detect possible goal misgeneralisation. This will be done through the implementation of two knowledge-based algorithms - the CN2 induction algorithm and the least-commitment search algorithm - in order to generate possible propositional descriptions of the intended goal. First, some preliminary work on interpretability is discussed, after which the two algorithms will be introduced, and finally we will implement these algorithms to generate descriptions of the objective for our three gridworld environments.

4.1 Explainability and Interpretability

While deep learning methods are very efficient and have been widely adopted over the past years, their performance often comes at the cost of transparency [51]. These systems are able to learn accurate classification or competent behavior over a wide range of applications, but are not able to provide a human-comprehensible explanation for their actions. As these models are scaled up, they are only expected to become more incomprehensible and opaque. Because of this, there is a growing demand for methods that can help us understand why these models make certain decisions.

Two concepts often used for this purpose is explainable AI (XAI) and machine interpretability. Although these concepts have been widely used, no formal definition for either concept exists [52]; while some attempts have been made, these definitions lack the mathematical rigor to use as a clear metric.

More widely embraced descriptions do not define a strict measure on AI systems, but simply attempt to capture the intention of what we would like these methods to achieve; one of the more popular definitions describes interpretability as “the degree to which a human can understand the cause of a decision” [53] [8].

There are several ways to make a black-box model more interpretable, such as making the model itself more interpretable by studying it and figuring out how exactly it works [54]; this is the case with the method of mechanistic interpretability mentioned in the introduction. Another method is the post-hoc use of interpretable algorithms to generate explain behavior of the black-box model; this is the approach we will take in the rest of the section.

4.2 Logical interpretability: rule induction algorithms

Symbolic algorithms can be used to explain the output of an opaque model by analysing this output and capturing this behavior in interpretable explanations. One way this is done is by generating a list of rules, which capture the bevavior of an agent to a degree of accuracy. One way to generate such a set of rules is with rule induction [55]; this is the method of generating logical rules which use the value of certain features to predict the value of an outcome variable.

This is relevant method in and of itself for classification tasks in the domain of supervised learning, but we can also use it as an interpretability tool for RL envrionments: namely, to use features from the environment to predict which action an agent will take.

[8]for the remainder of this section, we will simply use ”interpretability” to refer to this concept.

This method has been applied in order to synthesize a rule-based agent from observing an RL agent. After training an RL agent in this environment, data of its performance was collected and used to classify the agent's behavior depending on its observation, resulting in an interpretable rule-based agent which had a performance comparable to the original RL agent [56].

For our purposes, we do not want to generate interpretable explanations for the agent's policy, but instead generate descriptions of the intended objective as captured in the reward function: is it possible to classify when the environment's objective has been achieved, based on features from the environment which the agent can observe? If this can be done, we can generate descriptions of the objective which are consistent with the agent's behavior; with this, we can see what objectives are consistent with the agent's observations and obtained rewards, which correspond to objectives that the agent might generalise to.

4.2.1 Data collection for rule induction

Interpretable rules are generated using *selectors*, which are feature-value pairs. A dataset is generated by observing the agent interacting with the environment, and for each chosen feature, recording the selectors of each state. Since for each state in the environment, it is easy to check if a selector holds - that is, if the feature has the corresponding value in that state - induction rule algorithms can generate logical sentences with these selectors, which are either true or false corresponding to the outcome value this rule predicts.

4.2.2 CN2

The CN2 algorithm was first introduced in 1989 as an improved version of other rule induction algorithms [57]; This algorithm takes as input a dataset of features and outcomes, and generates an ordered ruleset which classifies new examples based on its feature values. Rules consist of conjunctions of selectors, and can be formulated as
"IF FEATURE1 = VALUE1 AND FEATURE2 = VALUE2 AND ... THEN RESULT = ..."

Candidate rules are found using beam search, and are rated according to a user-defined metric. For noisy data, often entropy and significance are used as a search heuristic; when a best rule is found, all examples covered by this rule are removed from the dataset and the process is repeated for the remaining examples.

For our purposes, we will need a different metric, since our data contains no noise; because the outcome we measure is whether the environment is completed, this result will always be accurate and consistent. If we were to use a metric like entropy, all rules would have the same ranking of zero entropy, and an arbitrary rule would be picked. Instead, we will use rule coverage; the more examples a rule correctly classifies, the better it is rated. This results in the most general - though still consistent - rules being picked, and avoids long lists of overly specific rules.

4.2.3 Version space learning: Least-commitment search

While it is interesting to see which objective the CN2 algorithm will predict, a downside is that it only generates one possible ruleset that predicts whether the objective is achieved or not. However, the problem we are faced with with GMG is that there are several objectives compatible with the agent's behavior on the training data. So, instead of generating just one explanation, ideally we would like to generate *all* possible hypotheses for the objective [9]. This approach is called version space learning [58]; given some space of possible hypotheses and a dataset of observations, we would like to iteratively refine this space by removing all hypotheses which are not consistent with the observed data, until only the valid hypotheses remain.

However, iterating over all possible feature configurations to check if they are consistent with the objective outcome is infeasible in practice. Even for the simple gridworlds, the number of features can quickly grow very large; if we would like to use the position of 6 objects as features in a 4×4 gridworld (for example, agent, target and colors), the size of the hypothesis space is already around 24 million [10] selector combinations.

The least-commitment search algorithm employs an idea to significantly cut down on the number of hypothesis that needs to be considered; the main idea behind the this algorithm is that there exists a partial order on this hypothesis space, based on whether some hypothesis is strictly more specific than another. For example, the hypothesis "AGENT POSITION = (2,3) AND TARGET POSITION = (0,2)" is more specific than "AGENT POSITION = (2,3)". If the latter is not always true for a completed environment, then certainly the former is also not true and thus can also not be a valid hypothesis. In fact, *every* hypothesis containing the selector "AGENT POSITION = (2,3)" can be discarded. On the other hand, if the former hypothesis also holds for non-complete states, this rule is *too general* to be a consistent hypothesis, and every hypothesis more general than this -such as the latter hypothesis - can also be discarded . Using this partial ordering, we can represent the set of all hypothesis consistent with the episode data with the sets of most general and most specific hypotheses still consistent with the data; these are called the "G-set" and "S-set" respectively. Any hypotheses which fall between these borders is consistent with the data, everything outside of it is inconsistent.

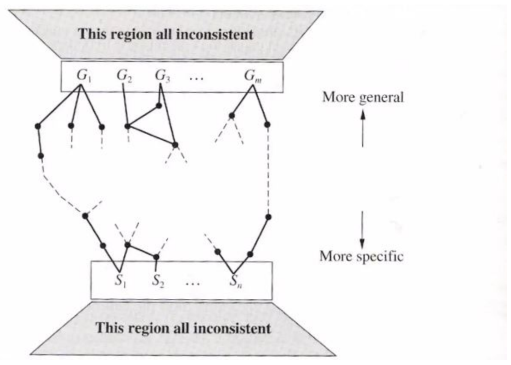
This region all inconsistent
G_1 G_2 G_3 ... G_m
More general
More specific
S_1 S_2 ... S_n
This region all inconsistent

Figure 12: A visualisation of the G-set and S-set as hypothesis boundaries [58]. Nodes represent hypotheses, edges represent the relationship of one hypothesis being more specific than the other.

[9]From here on, we will use "hypothesis" to mean a rule-based description to predict a binary outcome; for the rest of this thesis, the hypothesis will always be conjunctions of feature values, and the outcome will be whether the objective is completed or not

[10]Each object has 16 positions, and each feature could be excluded from a hypothesis, taking no value, for a total of 17^6 possibilities.

The least-search algorithm iterates over examples, and updates these boundary sets as necessary. If a hypothesis in either set is consistent with the example, it can remain in the set. If it is inconsistent, the example is either a false positive or false negative for the hypothesis. If the example is a false positive, this means the hypothesis predicts the objective has been achieved in this example, even though it hasn't; in this case, the hypothesis is too general, and should be made more specific by adding more selectors. If the example is a false negative (the hypothesis predicts the goal has not been achieved, even though it has), the hypothesis should be made more general by dropping selectors. In either of these cases, the corresponding set needs to be modified.

If an example is a false positive for a hypothesis in the G-set, the hypothesis is too general; it needs to be made more specific in order to be consistent with the example, so it is replaced by all its specialisations, generated by adding any possible selector to the hypothesis. If an example is a false negative, the hypothesis is too specific; however, since it is in the G-set, it cannot be generalised more without becoming inconsistent, and should thus be discarded from the set.

For hypotheses in the S-set, a false positive means the hypothesis is too general, and thus must be discarded from the S-set; for a false negative, the hypothesis is too specific, and is replaced by all its generalisations, which are generated by dropping any selector from the hypothesis.

After running over all examples, there are three possible outcomes; either the G-set and S-set are empty, meaning there are no consistent hypotheses based on the given features; there are several hypotheses left in the version space, meaning multiple hypotheses are consistent with the data; or - in our case ideally - there is only one hypothesis left; this is the only hypothesis that is consistent with the observed data. It should be noted that this is then the only consistent hypothesis based on the features present in the data; if more features are considered, more consistent hypotheses might be generated.

A general downside of this method is that the hypothesis space can easily collapse if there is noisy data present, since relevant hypotheses can immediately be discarded because of an incorrectly-labeled data point: However, even though the agent's learned policy might be noisy at times if it exhibits capability failure, the environment is always accurate in determining when the episode is completed, so there is no noise present in data classifying an episode as completed or not.

4.3 Applying rule induction algorithms to the GridWorld environments

Having implemented these algorithms, we can apply them to our gridworld environments in order to generate descriptions of the intended objective based on observation from the agent interacting with the environment. First, we collect relevant data from each environment, which we feed into both induction algorithms. The CN2 algorithm only returns one ruleset; we discuss why this rule in particular is chosen by the algorithm, and how we can obtain alternative rulesets. The least-commitment search algorithm returns a representation for all possible description hypotheses; we discuss how this representation can be used to find novel environment states which invalidate some of these hypotheses, and reduce the possibility of GMG.

4.3.1 Data collection from trained agents

For each gridworld, we select some relevant features of the gridworld from which we wish to generate hypotheses; which features specifically are chosen is mentioned below. Next, we run our trained agent for 1000 episodes in each environment and collect data on these features, along with whether the agent has completed the objective or not. This dataset is then used as the input for both rule induction algorithms.

Target gridworld features
Agent position: the location of the agent, in (x,y) coordinates
Targe positiont: the location of the target
Target distance: The Manhattan distance from the agent to the target

Property gridworld features
Target distance: the same as in the Target gridworld
Property distance 1 through 5: The distance to each of the indiviual properties

Chest gridworld features
Chest distance: The distance from the agent to the chest
Key distance: The distance from the agent to the key

Note that for the property gridworld, we only include distance to the target and properties, and no positions of objects; this is because, even for this very simple environment, including positions caused the boundary S-set update to increase significantly in size, since very false positive replaces a hypothesis with the n more general ones, where n is the number of features. For more on the efficiency and algorithmic complexity of this algorithm, see the discussion.

4.3.2 Results of CN2

After collecting our data, we pass it as input to our implementation of the CN2 algorithm. The resulting rulesets are as follows:

TargetGridWorld
IF Agent position IS (3, 3) THEN done IS True
DEFAULT: done IS False

This ruleset corresponds to the misgeneralised goal G_{mis}; it is only concerned with the position of the agent, not with the position of the target. This gives valuable information about the possiblity of misgeneralisation; while this does not guarantee that the agent will always try to optimize G_{mis}, it does show the fact that this objective is consistent with the observed data, and thus that an agent optimizing for this objective would be maximally rewarded on this subset of the training data.

The next question to consider is the following: why did the algorithm return this ruleset, and not a ruleset corresponding with, for example the intended objective? It turns out the answer is quite simple; since every misgeneralised objective is by definition indistinguishable from the intended goal based on observed behavior on the training data, the CN2 algorithm does not rank the rulesets corresponding to these objectives as higher or lower than the others. All of these rulesets have the same ranking, so the best ruleset that the algorithm keeps track of is the first one that is generated; this happens to be the rule containing the selector which appears first in the data, and this is a completely arbitrary property.

If we restructure the dataset by switching two colomns, resulting in the Target distance feature to appear first, CN2 returns the following ruleset:

IF TARGET DISTANCE IS 0 THEN DONE IS TRUE
DEFAULT: DONE IS FALSE

This is the intended objective. So, because these rules have identical coverage over the observed data, both rules are ranked the same according to CN2, and which one gets picked comes down to which attribute appears first in the dataset. For more on this, see the discussion.

Property gridworld
For the property gridworld, we get the following ruleset:

IF TARGET DISTANCE IS 0 THEN DONE IS TRUE
DEFAULT: DONE IS FALSE

This corresponds to the intended objective of the environment. However, just as with the target gridworld, the reason that this ruleset is chosen again depends only on the structure of the dataset, and not on some specific property of the target distance itself. If we repeat the process of swapping two columns in the dataset such that the property1 distance is the first feature we record, we get the following ruleset:

IF PROPERTY1 DISTANCE IS 0 THEN DONE IS TRUE
DEFAULT: DONE IS FALSE

So again, which rule CN2 returns is completely arbitrary and based on in what order the data are collected.

Chest gridworld

For the final environment, running the algorithm on the collected data results in the following ruleset:

IF CHEST DISTANCE IS 0 THEN CLASS IS TRUE
DEFAULT DONE IS FALSE

This is an interesting result, since behavior based on this ruleset is not rewarded during training; the agent needs to move to the key first, so simply moving to the chest will not give any reward.

However, CN2 still returned this as a possible objective, since the episode was completed every single time the agent reached the chest. Because the agent has optimised its behavior and only moves to the chest if it has gotten the key, the training data did not contain any examples where the agent is at the chest without having the key. As this is not in the training data, it the induction algorithm cannot infer that this is insufficient to complete the environment, and thus returns it as a valid possible objective - even if the agent has already learned that this is not a viable strategy.

We can easily remedy this by including some random agent behavior in the training data, such that it contains examples of the agent being at the chest without a key. With these observations, the algorithm returns the following ruleset:

IF Key Distance IS 0 AND Chest Distance IS 0 THEN done IS True
DEFAULT done IS False

This classification corresponds to a correct objective, in this case the intended objective. However, with the current features, it is not possible to construct a ruleset for the environment with multiple keys. Since the "Key distance" feature corresponds to the distance to only one of the two keys, it is not a sufficient feature to classify succes in this environment, because the environment can be completed with the other keys as well. In order to generate suitable rules for this environment, we need to introduce two new features: these are "Any key" and "All keys", measuring if the agent has obtained either of the keys and both of the keys respectively; with these features, we can construct rulesets corresponding to G_{int} and G_{mis} respectively. With these features, we obtain a ruleset again corresponding to G_{int}:

IF Any key IS True AND Chest Distance IS 0 THEN CLASS IS True
DEFAULT done IS False

It has become apparent that the choice of features play a deciding role in wether relevant rulesets can be generated. Because of this, the validity of this method to detect GMG is questionable, since alternative hypotheses for the objective can only be detected if corresponding features are measured, which requires some prerequisite intuition of how an agent might misgeneralise; for more, see the discussion.

4.3.3 Results of least-commitment search

For the least-commitment search algorithm, the structure of the data no longer matters, since the algorithm generates all consistent hypotheses at once. With this, we can immediately see if GMG is possible on the basis of the features we have measured.

Target gridworld
When running the algorithm on data gathered from the Target gridworld, we get the following results:
G-set hypotheses:
Agent position = (3, 3)
Target distance = 0
Target position = (3, 3) and Agent position = (3, 3)

TARGET POSITION = (3, 3) AND TARGET DISTANCE = 0
S-set hypotheses:
AGENT POSITION = (3, 3) AND TARGET POSITION = (3, 3) AND TARGET DISTANCE = 0

From the G-set, we can immediately see that several hypotheses are consistent with the gathered data, even though not all hypotheses are reflected by the reward function. In this case, it is clear that several hypotheses need to be eliminated in order to remove the possibility of GMG; this would mean that for two hypotheses, it should be investigated whether an episode is complete if one is true and the other isn't. For example, it can be investigated what happens if AGENT POSITION = (3, 3) but TARGET DISTANCE $\neq 0$ This can only be the case if the target is not at position (3,3) [11].

When we include training data where the target can be at any position, we get the following boundary sets:

G-set hypotheses:
TARGET DISTANCE = 0
S-set hypotheses:
TARGET DISTANCE = 0

Since both sets contain the same single element, we can conclude that this is the only hypothesis consistent with the training data. Thus, we can conclude that no misgeneralisation can occur on the basis of these features. However, misgeneralisation may still occur on the basis of any other features not considered in the collected data; just as with the CN2 algorithm, our choice of features largely determines in which scope we are able to detect the possibility of GMG.

Property gridworld
For data from the Property gridworld, we get the following results:

G-set hypotheses:
TARGET DISTANCE = 0
PROPERTY1 DISTANCE = 0
PROPERTY2 DISTANCE = 0
PROPERTY3 DISTANCE = 0
PROPERTY4 DISTANCE = 0
PROPERTY5 DISTANCE = 0
S-set hypotheses:
TARGET DISTANCE = 0 AND PROPERTY1 DISTANCE = 0 AND PROPERTY2 DISTANCE = 0 AND PROPERTY3 DISTANCE = 0 AND PROPERTY4 DISTANCE = 0 AND PROPERTY5 DISTANCE = 0

Again, there are several hypotheses, all of which are consistent with the observed data. Note that even though there are many possible hypotheses, none has the ability to accurately predict the agent's behavior, namely, that the agent's tendency to go to a position is proportional to the number of properties in that position. However, the intended behavior can again be easiliy discovered if training examples are supplied where the different properties are separated; this will eliminate every

[11]Note that in some cases, two hypotheses can actually be identical in meaning, and thus cannot be separated by any example; for more on this, see the discussion in section 6.

hypothesis except for TARGET DIST = 0.

Chest gridworld Finally, we use data from the 1-key chest gridworld with the "Any key" and "All key" features to get the following results:

G-set hypotheses:
CHEST DISTANCE = 0
ANY KEY = TRUE AND CHEST DISTANCE = 0
ALL KEYS = TRUE AND CHEST DISTANCE = 0
S-set hypotheses:
CHEST DISTANCE = 0 AND ANY KEY = TRUE AND ALL KEYS = TRUE

Again, the first G-set hypothesis is incorrect, but can be removed by supplementing the dataset with non-optimal behavior. The other hypotheses are as expected, and correspond to G_{int} and G_{mis}.

If we run the algorithm on the 2-key environment, we get the following sets:

G-set hypotheses:
CHEST DISTANCE = 0 AND ANY KEY = TRUE
S-set hypotheses:
CHEST DISTANCE = 0 AND ANY KEY = TRUE

This representation of the hypothesis space only contains one hypothesis, namely the one corresponding to the intended objective. This is because the agent did not have a perfect misgeneralisation rate after training, such that it sometimes collected one key and sometimes both keys. Because of this, it is able to receive rewards for a more diverse set of outcomes, and the algorithm can discard the hypotheses where all keys are needed.

5 Discussion and Further Work

Given the results from section 3 and 4, there are several problematic aspects and open problems remaining; the most important ones are discussed here.

5.1 Features included in the induction algorithm

A problem with using features gathered from episode observations to define objectiv hypotheses, is that the induction algorithms can only formulate misgeneralised objectives on the basis of these chosen featurs. This heavily limits to what extent misgeneralised objectives can be extracted; it seems we need to predict what features the agent might misgeneralise on in order to generate these hypotheses, which somewhat defeats the purpose of these algorithms.

A possible solution to this is to not specify any features; we can simply take the entries of the agent's binary observation vector as features.
However, this approach has three problems of its own. First of all, the agent's observation entries might not always correspond to concepts that humans can easily interpret. For image recognition, it does not suffice to explain a labelling based on the color value of certain pixels; instead, higher-order features need to be constructed in order to generate understable explanations.

That brings us to the second problem: not all properties on which an agent might misgeneralise are directly encoded into the observation. For example, in the gridworlds we use distance from the agent to an object as a property, but this is only indirectly represented in the agent's observation; there is no entry which corresponds to distance, but this property is dependent on other entries. So, using only the agent's observation entries would probably not result in valid hypotheses.

Lastly, it is simply infeasible to include that many features. Because the number of possible hypotheses with binary features is 3^n, with n number of features (every feature is either true, false or omitted from the hypothesis), the version space that the least-commitment algorithm needs to search becomes enormous with a large number of features. Even if the algorithm need not consider every hypothesis in this space, the relevant number of hypotheses still increases exponentially as the number of features increases.

5.2 Terminal goal states as objective

The rule induction algorithms that were used made rules to predict certain outcomes; in our case, the outcome was whether the environment has been completed or not. This method is clearly potentially useful for environments where the objective is to reach some terminal goal state, since reaching this goal state is strictly positive, while any other state can be classified as strictly negative.

However, not every environment can be formulated in this way; environments where the objective is for the agent to avoid certain states, or simply survive in the environment for as long as possible, do not have clearly defined goal states. As such, it is less feasible to classify any observation as succesful or not, which hampers our ability to formulate hypotheses that describe success

Further research could be conducted on predicting reward, instead of having completed the environment; this would enable hypotheses generation on a wider range of environments without a predefined goal state, but simply more and less preferred states. However, this limits which induction algorithms can be used; while the CN2 algorithm can predict several outcome values, the least-commitment search algorithm can only classify binary outcomes, and as such is not suitable for this task.

5.3 Incomplete data from optimal behavior

If the agent behaves optimally towards some goal in its environment, this may actually result in more hypotheses being generated from observation data than if observing a random agent; this is because the optimal agent competently avoids certain states that are not rewarded, and so these cannot be recorded in observation data as indeed having no reward. However, an upside of observing agent behavior is that we will often be able to record success as well; only observing a random agent in an environment will lead to very sparse data on what constitutes success, since the goal state will almost never be reached.

An optimal dataset would contain a mixture of both optimal and random behavior, to reap from both benefits. A possible way to implement this is to record data during the course of training, instead of after completing training; this way, random behavior is recorded at first, and is gradually supplemented by optimal behavior as the agent learns correct behavior.

5.4 Distinguishing hypotheses

An unresolved problem is that although our rule induction algorithms can generate multiple hypotheses, it is not immediately clear if these hypotheses could differ on OOD data, or if they are objectively identical. For example, the least-commitment search algorithm applied to the chest grid-world will generate the hypotheses ALL KEYS = TRUE AND CHEST DISTANCE = 0 and ALL KEYS = TRUE AND ANY KEY = TRUE AND CHEST DISTANCE = 0. While these hypotheses differ in description, they are actually identical, since the agent having all keys logically implies that it has some key [12].

Further research could be conducted on verifying whether for two hypotheses, there exists a state in the environment which seperates these hypotheses, meaning that only one hypothesis is consistent with this state. If such a state is found, the corresponding reward can be observed and one of the hypotheses discarded. If no such state exists, the hypotheses are truly identical within the environment.

[12]given, of course, that there is at least one key present in the environment.

Conclusion

This work has studied the properties of goal misgeneralisation by designing simple environments and training agents in this environment which exhibit misgeneralised behavior. It is shown that very little is needed for goal misgeneralisation to occur; as long as the training architecture uses function approximation, and there is ambiguity in the observed training data and corresponding rewards - that is, if there are multiple objectives whose corresponding behavior are similar on the observed training data, but differ outside this data - GMG can occur, even with minimal training time. Whether this will occur, and to what extent, depends on how likely the agent is to optimize on the unintended goal.

For the target Gridworld, the corresponding agent robustly optimizes on positional and directional proxies of the objective instead of attributes of the objective present in the agent's observation, showing that these proxies are strong possible indicators of possible GMG. Removing these proxies, the strength of features present in the agen's observation also influences behavior, as shown with the property gridworld; which of two options the agent pursues is shown to correlate with how strongly this option is encoded in the agent's representation.

Rule induction algorithms were tested for their ability to predict whether it is possible for an agent to optimize on an unintended objective, based on what is consistent with the agent's behavior in the training environment. While the implemented CN2 algorithm was able to generate a consistent objective hypothesis, it was only able to generate a single hypothesis; which hypothesis was chosen depended only on arbitrary properties of the structure of the collected data. The least-commitment search algorithm was able to generate all feature-based hypotheses consistent with the observed data, but the scope of these hypotheses is severely limited by which features are chosen with which to generate the hypothesis space; because of this, misgeneralised behavior may not always be predicted with these algorithms, as is shown with the chest gridworld.

With these results, the practical use of these rule induction algorithms is shown to be very limited, mainly because of two reasons: considered features need to be specified beforehand, which limits the scope of hypotheses that the algroithms can search through. Second, the runtime of these algorithms vastly increases along with the number of features and corresponding selectors, rendering it impractical to deploy the algorithms in more complex situations where the number of features to consider can be vast.

Although these methods give clear representations of possible ways of GMG, the limitations of these methods probably outweigh their practicality for everything past toy environments. While these methods might still remain useful in toy environments to research possible novel ways GMG can occur, they have little practical use for predicting and preventing GMG in complex, real-world systems.

References

[1] I. Gabriel, "Artificial intelligence, values, and alignment," *Minds and machines*, vol. 30, no. 3, pp. 411–437, 2020.

[2] P. Eckersley, "Impossibility and uncertainty theorems in ai value alignment (or why your agi should not have a utility function)," *arXiv preprint arXiv:1901.00064*, 2018.

[3] A. Y. Ng, S. Russell, *et al.*, "Algorithms for inverse reinforcement learning.," in *Icml*, vol. 1, p. 2, 2000.

[4] A. Conmy, A. N. Mavor-Parker, A. Lynch, S. Heimersheim, and A. Garriga-Alonso, "Towards automated circuit discovery for mechanistic interpretability," *arXiv preprint arXiv:2304.14997*, 2023.

[5] N. Nanda, L. Chan, T. Liberum, J. Smith, and J. Steinhardt, "Progress measures for grokking via mechanistic interpretability," *arXiv preprint arXiv:2301.05217*, 2023.

[6] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, *et al.*, "Mastering the game of go with deep neural networks and tree search," *nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[7] M. F. A. R. D. T. (FAIR)†, A. Bakhtin, N. Brown, E. Dinan, G. Farina, C. Flaherty, D. Fried, A. Goff, J. Gray, H. Hu, *et al.*, "Human-level play in the game of diplomacy by combining language models with strategic reasoning," *Science*, vol. 378, no. 6624, pp. 1067–1074, 2022.

[8] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[9] OpenAI, "Gpt-4 technical report," *ArXiv*, vol. abs/2303.08774, 2023.

[10] S. Russell, D. Dewey, and M. Tegmark, "Research priorities for robust and beneficial artificial intelligence," *Ai Magazine*, vol. 36, no. 4, pp. 105–114, 2015.

[11] "Pause giant ai experiments: An open letter," May 2023.

[12] "Statement on ai risk," May 2023.

[13] N. Wiener, "Some moral and technical consequences of automation: As machines learn they may develop unforeseen strategies at rates that baffle their programmers.," *Science*, vol. 131, no. 3410, pp. 1355–1358, 1960.

[14] A. Turing, "Can digital computers think? typescript with annotations of a talk broadcast on bbc third programme, 15 may," tech. rep., AMT/B/5. In [39]. URL: http://www.turingarchive.org/browse.php/B/5, 1951.

[15] S. Russell, "Human-compatible artificial intelligence," *Human-Like Machine Intelligence*, pp. 3–23, 2021.

[16] J. Dressel and H. Farid, "The accuracy, fairness, and limits of predicting recidivism," *Science advances*, vol. 4, no. 1, p. eaao5580, 2018.

[17] B. Christian, *The alignment problem: Machine learning and human values.* WW Norton & Company, 2020.

[18] J. Buolamwini and T. Gebru, "Gender shades: Intersectional accuracy disparities in commercial gender classification," in *Conference on fairness, accountability and transparency*, pp. 77–91, PMLR, 2018.

[19] N. Grant and K. Hill, "Google's photo app still can't find gorillas. and neither can apple's.," May 2023.

[20] T. Bolukbasi, K.-W. Chang, J. Y. Zou, V. Saligrama, and A. T. Kalai, "Man is to computer programmer as woman is to homemaker? debiasing word embeddings," *Advances in neural information processing systems*, vol. 29, 2016.

[21] G. F. Cooper, V. Abraham, C. F. Aliferis, J. M. Aronis, B. G. Buchanan, R. Caruana, M. J. Fine, J. E. Janosky, G. Livingston, T. Mitchell, *et al.*, "Predicting dire outcomes of patients with community acquired pneumonia," *Journal of biomedical informatics*, vol. 38, no. 5, pp. 347–366, 2005.

[22] S. Wachter, B. Mittelstadt, and L. Floridi, "Why a right to explanation of automated decision-making does not exist in the general data protection regulation," *International Data Privacy Law*, vol. 7, no. 2, pp. 76–99, 2017.

[23] A. Selbst and J. Powles, ""meaningful information" and the right to explanation," in *conference on fairness, accountability and transparency*, pp. 48–48, PMLR, 2018.

[24] R. Sutton, "The bitter lesson," *URL http://www. incompleteideas. net/IncIdeas/BitterLesson. html*, 2019.

[25] J. McCarthy, M. L. Minsky, N. Rochester, and C. E. Shannon, "A proposal for the dartmouth summer research project on artificial intelligence, august 31, 1955," *AI magazine*, vol. 27, no. 4, pp. 12–12, 2006.

[26] J. Steinhardt, "Ai forecasting: One year in," *LessWrong. Available online at: https://www.lesswrong.com/posts/CJw2tNHaEimx6nwNy/ai-forecasting-one-year-in (accessed June 31, 2023)*, 2022.

[27] V. C. Müller and N. Bostrom, "Future progress in artificial intelligence: A survey of expert opinion," *Fundamental issues of artificial intelligence*, pp. 555–572, 2016.

[28] K. Grace and J. Salvatier, "2016 expert survey on progress in ai," *AI Impacts. Available online at: https://aiimpacts.org/2016-expert-survey-on-progress-in-ai/ (accessed December 7, 2022)*, 2016.

[29] Z. Stein-Perlman, B. Weinstein-Raun, and K. Grace, "2022 expert survey on progress in ai," *AI Impacts. Available online at: https://aiimpacts. org/2022-expert-survey-on-progress-in-ai (accessed December 7, 2022)*, 2022.

[30] A. Cotra, "Forecasting tai with biological anchors. 2020," *URL https://docs. google. com/document/d/1IJ6Sr-gPeXdSJugFulwIpvavc0atjHGM82QjIfUSBGQ/edit.*

[31] S. Bubeck, V. Chandrasekaran, R. Eldan, J. Gehrke, E. Horvitz, E. Kamar, P. Lee, Y. T. Lee, Y. Li, S. Lundberg, *et al.*, "Sparks of artificial general intelligence: Early experiments with gpt-4," *arXiv preprint arXiv:2303.12712*, 2023.

[32] N. Bostrom, "The superintelligent will: Motivation and instrumental rationality in advanced artificial agents," *Minds and Machines*, vol. 22, no. 2, pp. 71–85, 2012.

[33] S. M. Omohundro, "The basic ai drives," in *AGI*, vol. 171, pp. 483–492, 2008.

[34] R. Ngo, "The alignment problem from a deep learning perspective," *arXiv preprint arXiv:2209.00626*, 2022.

[35] J. Carlsmith, "Is power-seeking ai an existential risk?," *arXiv preprint arXiv:2206.13353*, 2022.

[36] A. M. Turner, L. Smith, R. Shah, A. Critch, and P. Tadepalli, "Optimal policies tend to seek power," *arXiv preprint arXiv:1912.01683*, 2019.

[37] A. Turner and P. Tadepalli, "Parametrically retargetable decision-makers tend to seek power," *Advances in Neural Information Processing Systems*, vol. 35, pp. 31391–31401, 2022.

[38] V. Krakovna and J. Kramar, "Power-seeking can be probable and predictive for trained agents," *arXiv preprint arXiv:2304.06528*, 2023.

[39] I. Popov, N. Heess, T. Lillicrap, R. Hafner, G. Barth-Maron, M. Vecerik, T. Lampe, Y. Tassa, T. Erez, and M. Riedmiller, "Data-efficient deep reinforcement learning for dexterous manipulation," *arXiv preprint arXiv:1704.03073*, 2017.

[40] T. M. VII, "The first level of super mario bros. is easy with lexicographic," 2013.

[41] J. Randløv and P. Alstrøm, "Learning to drive a bicycle using reinforcement learning and shaping.," in *ICML*, vol. 98, pp. 463–471, 1998.

[42] A. Y. Ng, D. Harada, and S. Russell, "Policy invariance under reward transformations: Theory and application to reward shaping," in *Icml*, vol. 99, pp. 278–287, Citeseer, 1999.

[43] S. Thulasidasan, S. Thapa, S. Dhaubhadel, G. Chennupati, T. Bhattacharya, and J. Bilmes, "An effective baseline for robustness to distributional shift," in *2021 20th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pp. 278–285, IEEE, 2021.

[44] F. S. Melo and M. I. Ribeiro, "Q-learning with linear function approximation," in *Learning Theory: 20th Annual Conference on Learning Theory, COLT 2007, San Diego, CA, USA; June 13-15, 2007. Proceedings 20*, pp. 308–322, Springer, 2007.

[45] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[46] L. L. Di Langosco, J. Koch, L. D. Sharkey, J. Pfau, and D. Krueger, "Goal misgeneralization in deep reinforcement learning," in *International Conference on Machine Learning*, pp. 12004–12019, PMLR, 2022.

[47] R. Shah, V. Varma, R. Kumar, M. Phuong, V. Krakovna, J. Uesato, and Z. Kenton, "Goal misgeneralization: Why correct specifications aren't enough for correct goals," *arXiv preprint arXiv:2210.01790*, 2022.

[48] T. Benson-Tilsen and N. Soares, "Formalizing convergent instrumental goals.," in *AAAI Workshop: AI, Ethics, and Society*, 2016.

[49] A. Paszke, S. Gross, F. Massa, A. Lerer, and J. P. Bradbury, "An imperative style, high-performance deep learning library," *Advances in Neural Information Processing Systems*, vol. 32.

[50] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[51] E. Puiutta and E. M. Veith, "Explainable reinforcement learning: A survey," in *Machine Learning and Knowledge Extraction: 4th IFIP TC 5, TC 12, WG 8.4, WG 8.9, WG 12.9 International Cross-Domain Conference, CD-MAKE 2020, Dublin, Ireland, August 25–28, 2020, Proceedings 4*, pp. 77–95, Springer, 2020.

[52] A. Adadi and M. Berrada, "Peeking inside the black-box: a survey on explainable artificial intelligence (xai)," *IEEE access*, vol. 6, pp. 52138–52160, 2018.

[53] T. Miller, "Explanation in artificial intelligence: Insights from the social sciences," *Artificial intelligence*, vol. 267, pp. 1–38, 2019.

[54] P. Linardatos, V. Papastefanopoulos, and S. Kotsiantis, "Explainable ai: A review of machine learning interpretability methods," *Entropy*, vol. 23, no. 1, p. 18, 2020.

[55] J. W. Grzymala-Busse, "Rule induction," *Data mining and knowledge discovery handbook*, pp. 277–294, 2005.

[56] Y. Coppens, D. Steckelmacher, C. M. Jonker, and A. Nowé, "Synthesising reinforcement learning policies through set-valued inductive rule learning," in *International Workshop on the Foundations of Trustworthy AI Integrating Learning, Optimization and Reasoning*, pp. 163–179, Springer, 2021.

[57] P. Clark and T. Niblett, "The cn2 induction algorithm," *Machine learning*, vol. 3, no. 4, pp. 261–283, 1989.

[58] S. J. Russell and P. Norvig, *Artificial intelligence a modern approach.* Pearson Education, Inc., 2010.