Resolving Spurious Correlations in Causal Models of Environments via Interventions

Sergei Volodin, Nevan Wichers, Jeremy Nixon
Google Research
{volodin,wichersn,jeremynixon}@google.com

December 9, 2020

Abstract

Causal models bring many benefits to decision-making systems (or agents) by making them interpretable, sample-efficient and robust to changes in the input distribution. However, spurious correlations can lead to wrong causal models and predictions. We consider the problem of inferring a causal model of a reinforcement learning environment and we propose a method to deal with spurious correlations. Specifically, our method designs a reward function which incentivises an agent to do an intervention to find errors in the causal model. The data obtained from doing the intervention is used to improve the causal model. We propose several intervention design methods and compare them. The experimental results in a grid-world environment show that our approach leads to better causal models compared to baselines: learning the model on data from a random policy or a policy trained on the environment's reward. The main contribution consists of methods to design interventions to resolve spurious correlations.

1 Introduction

Reinforcement Learning (Sutton & Barto, 2018) (RL) is a general framework where an agent interacts with an environment to optimize a reward. The recent successes (Mnih et al., 2013; Silver et al., 2016; Berner et al., 2019; Vinyals et al., 2019; Akkaya et al., 2019) of applying RL algorithms to games sparked an interest in the framework as a general solution to Artificial Intelligence problems. However, current RL agents have significant shortcomings (Irpan, 2018; Kurenkov, 2018), such as lack of interpretability, lack of robustness to distributional shift, susceptibility to adversarial examples (Gleave et al., 2019), and high sample complexity.

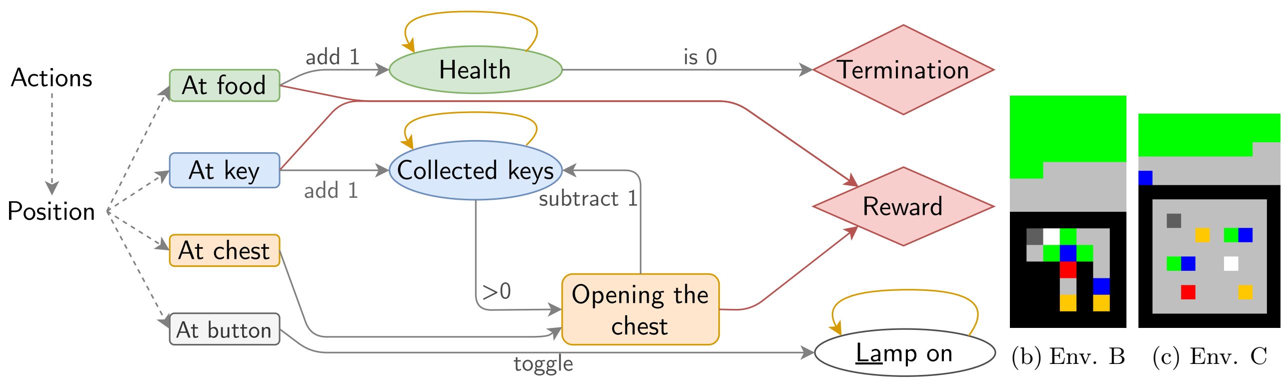
One of the frameworks that can arguably solve these issues is Causal Reasoning (Pearl & Mackenzie, 2018; Halpern & Pearl, 2005; Pearl, 2018). The framework considers a system whose states change with time. The goal is to model this system as a set of equations which predict the next state given the history.

One of the problems that causal reasoning considers is the inference of the model given limited data from a dynamical system (Javed et al., 2020). When solving this problem, spurious correlations are a major issue. They are pairs of variables which are associated given the data but not actually causally related. For example, we might care about predicting when the Sun rises. If we hear a rooster making a sound every time before the Sun rises, we might learn to rely on that feature. However, this would be a spurious correlation, since the sound does not actually *cause* the Sun to rise. If we move to an area where we cannot hear the rooster, we will know that it is the case: the Sun rises despite the fact that we do not hear the rooster.

To resolve this issue, we need to obtain a more diverse dataset to train the causal model. To do so, we need to perform an *intervention* (Halpern & Pearl, 2005) in the environment, such as moving to another location. In general, interventions are sets of actions that change the values of some variables in the causal model. They are used to create a better causal model. Data from an intervention can cause a spurious correlation to disappear from the dataset in a sense that it is no longer beneficial to rely on the spurious correlation. Therefore, a model trained on the new dataset would be forced to use more reliable features, for example, the current relative position of the Sun and the Earth.

Contribution. The main contribution consists of methods to design interventions to resolve spurious correlations in Causal Models of Reinforcement Learning environments (Section 3). In addition, we formulate the problem of learning a causal model of an RL environment mathematically in Section 2. The metric of success is how well we can uncover the correct causal model, rather than the final performance of the agent.

Method overview and steps. In this paper, we consider the causal model of the environment that the agent is deployed into. To do so, we train the model on the rollouts data obtained from agent-environment interaction. Our approach collects rollouts, then trains a causal model on that data, then computes the (approximate) best reward function to uncover spurious correlations. This reward depends on the causal model and encourages the agent to perform structured interventions in the environment. We thus call this reward



(a) Causal diagram for the KeyChest environment (b) Env. B (c) Env. C

Figure 1: Left 1a: The causal diagram of the environment which the agent should learn. The player needs to collect food and keys. Keys are used to open chests and the number of keys is displayed above the first black line on Figure 1c (and it is 0 in Figure 1b). Top row with health decreases at every time-step, and the episode ends if it is 0. The button toggles the lamp (black/white) which gives no reward. Orange arrows going to the left show that the variable depends on itself at the previous time-step. Red arrows indicate that the node results in a reward. All other arrows mean that the child depends on the value of the parent at the previous time-step, except for the node Opening the Chest, which depends on values at the current time-step. The right two figures 1b, 1c show the layouts of environments B and C. The agent is red, food is green, keys are blue, chests are orange, button is gray, and the lamp is either white or black (white on the figures). The dynamics of the environment is described in Section 4.

an *intervention reward*. We propose several methods calculate the intervention reward: from hand-designed heuristics to end-to-end learned methods. After training a policy for the intervention reward, we collect data from rollouts of the agent's policy. Then we train the causal model on this new data to learn a model with fewer spurious correlations. Then a new intervention is designed based on the new causal model to repeat the cycle.

We would like to understand the relationship between high-level concepts in the environment, such as positions of players, rather than low-level pixel representation. So, our causal model uses *computed features* rather than raw observations. In our setup, the features are computed from observation by a hand-designed function.

2 Problem

We focus on the problem of inferring a causal graph (causal model) of the environment in a reinforcement learning setting.

2.1 Notation

We use $[n]$ to denote the set of first n positive integers: $[n] = \{1, 2, 3, ..., n\}$. $x \sim X$ means sampling a random variable x from a distribution X.

Reinforcement learning. We consider the Reinforcement Learning setup consisting of two entities: the agent with a policy π and the environment μ. They interact with each other: the environment gives the agent an observation $o \in \mathcal{O}$, then the agent gives an action $a \in \mathcal{A}$ and the environment responds with a reward $r \in \mathcal{R}$. We call $x = (o, a, r)$ a step. After one step, the cycle is repeated. The environment begins and ends the interaction. We call one interaction an episode (or a *rollout*) with *history* $h = ((o_1, a_1, r_1), \dots, (o_T, a_T, r_T))$ where T is the *length* of the episode. Sometimes we call multiple episodes $(h_1, ..., h_k)$ a *history* as well. A *policy* π is a mapping from observations to a distribution over actions. By (μ, π) we denote the distribution of histories obtained from interactions between the environment μ and the policy π.

Causal models. We take the standard definition of Functional Causal models from Pearl (2009) (page 26). We consider a set of variables f_i which are evolving in time. The model is a directed graph G with variables as nodes. The value at the next time-step $t+1$ for a variable f_i is determined by a function of its parents $\mathbf{PA}(f_i)$ in the graph: $f_i^{t+1} = M_i(f_j^t,\ j \in \mathbf{PA}(f_i))$. Here M_i is a fixed but unknown function.

Causal learning from step features. We want to do causal learning on higher level features (Nachum et al., 2018) instead of the raw observations from the environment. Therefore, we extract high-level features from the observations using a hand designed function $f : \mathcal{O} \to \mathcal{F}$ with $\mathcal{F} = \mathbb{R}^F$, and do the causal learning

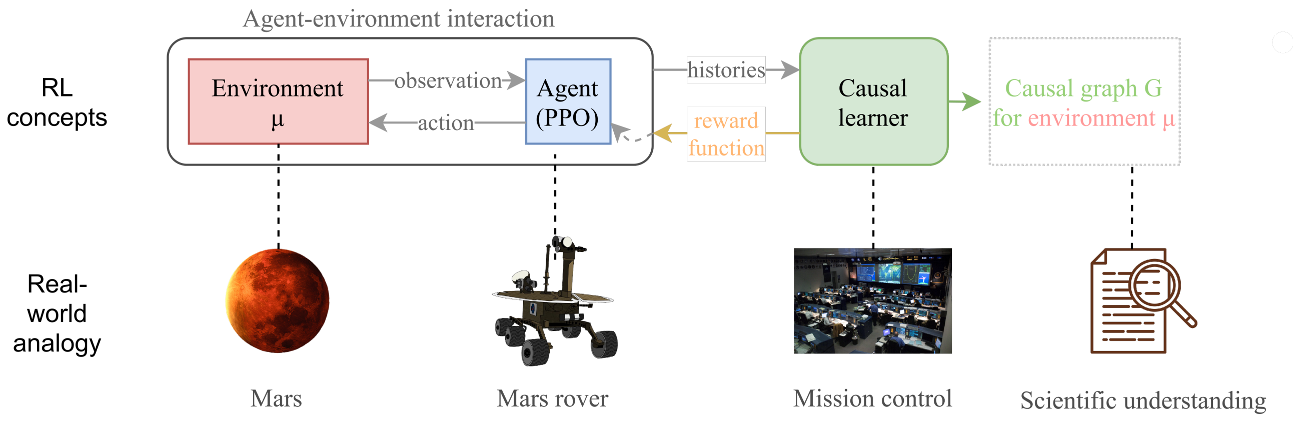


Figure 2: Problem setup. RL concepts (top row) are illustrated with real-world analogy (bottom row). Specifically, an environment μ is illustrated by a novel planet to explore, the agent corresponds to a rover on that planet executing low-level commands from the mission control, which corresponds to the causal learner that we introduce. At the end, the learner outputs a graph G which corresponds to high-level understanding of the planet or the environment μ

on those extracted features. From this point on, we use features/nodes/variables f_i interchangeably since they mean the same thing. Features for our setup are shown in Figure 1a.

We run a RL agent following a policy π in an environment, which generates a history h of observations $\mathcal{O}$. Then we compute features from those states using a hand-designed function f. Then, we learn a causal graph from that history by fitting the functions M_i for each node to predict features at the next time-step given the current features. In our setup, functions M_i are linear. In addition, to reduce the number of edges in the model, we regularize M_i for sparsity with an l_1 loss[1]. This method is called *Granger* causality (Granger, 1969).

To learn the model, we consider all of the nodes as potential parents for any given node. After learning the model, we threshold the weights based on their magnitude to get the parents $\mathbf{PA}(f_i)$ for each node: edges in the graph correspond to weights of high magnitude, and non-existent edges correspond to weights close to 0.

Mean squared loss of the causal model G on histories data generated by interaction (μ, π) between an environment μ and an agent running a policy π is defined as:

$$L_\pi(G) = \mathbb{E}_{h\sim(\mu,\pi)} \sum_{t=1}^{T} \sum_{i=1}^{F} \left(f_i^t - \hat{f}_i^t\right)^2 \tag{1}$$

Here $\tilde{f}_i^t = M_i(f_j^{t-1}, j \in [n])$ is the predicted value for feature i at time-step t by the causal model G in case if $t > 0$ or a constant in case of $t = 0$, the index t represents the episode step number, and the random variable T is the number of steps in the episode.

This loss measures how well the model can predict the features of the next time-step f_t given previous features f_{t-1}. This is simply the loss of linear regression when fitting functions M_i on the histories data. Note that $L = 0$ corresponds to the perfect fit between the model and the true environment dynamics.

For a finite set of histories $\{h_e\}_{e=1}^{E}$ consisting of E episodes sampled from (μ, π), we define the loss of the causal model G as:

$$L_{\pi,E}(G) = \frac{1}{E} \sum_{e=1}^{E} \sum_{t=1}^{T_e} \sum_{i=1}^{F} \left(f_i^{e,t} - \tilde{f}_i^{e,t}\right)^2 \tag{2}$$

Here e is the episode number, t is the step index in the episode, and T_e is the number of steps in the episode with index e. Index i indicates the feature, and F is the number of features. $f_i^{e,t}$ thus is the feature i at the time-step t of episode e.

By the Central Limit Theorem, $L_\pi(G) = \lim_{E\to\infty} L_{\pi,E}(G)$.

Spurious correlations and interventions. To sum up, we would like to design an algorithm to find a graph G which fits the environment μ, without giving the algorithm the ground truth graph G^* but only the ability to interact with the environment. To do so, we can minimize the loss $L_{\pi,E}(G)$ over G given histories generated by some policy π. For that, we run a policy π in the environment to collect history h and then we minimize L on the histories.

Assuming we had a learning algorithm that could converge to $L_\pi(G)$ for any π, our Proposition 1 (in the Appendix) shows that a random (uniform over actions) policy π_r is sufficient to learn the graph in a realizable case defined by Shalev-Shwartz & Ben-David (2014) where some graph G^* perfectly fits the data[2]. This happens

[1]Sparse causal models of dynamical systems for interpretability can be seen in (Bengio, 2017)
[2]Note that in practice, a random policy might take too much time to explore the environment

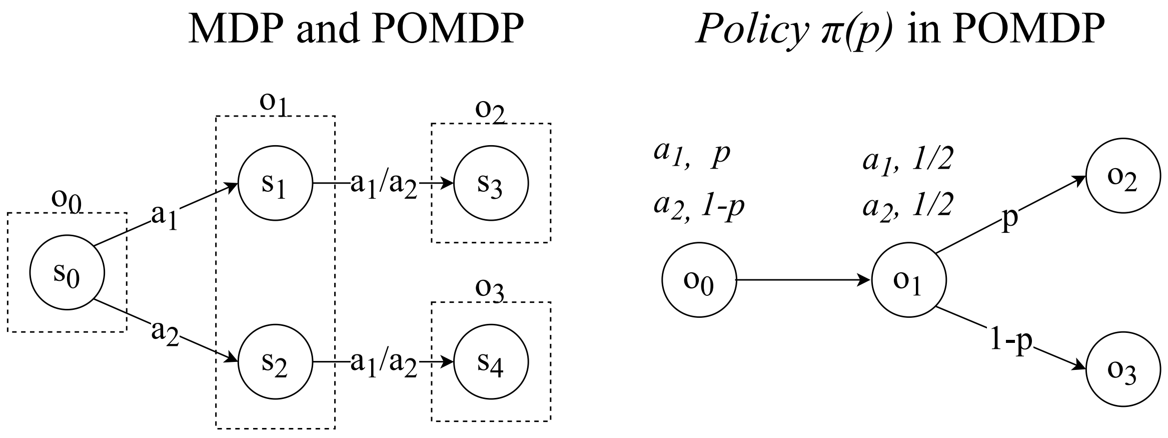


Figure 3: Environment with no ground truth graph G. In this example, the optimal graph depends on the policy collecting data. Specifically, given the observation o_1, the agent's prediction on what happens next depends on the action taken in o_0, because the true state (either s_1 or s_2) is inaccessible. The causal graph in this case depends on the policy's actions in state o_0 because by our assumption, we only use single time-step dependencies. **Full description:** On the left, the state MDP (Markov Decision Process) is shown together with groupings of states resulting in equal observations. On the right, the resulting POMDP (Partially-Observable Markov Decision Process) after the grouping is shown, along with an example policy with parameter p, $\pi(p)$. In the MDP, there are 2 actions and 5 states: s_0, s_1, s_2, s_3, s_4. In s_0 (initial state) if we take a_1, we go to s_1. If we take a_2, we go to s_2. From s_1 and s_2, it does not matter which action is taken, as there is only one transition (to s_3 and s_4 respectively). s_3 and s_4 are terminal states. These states are grouped into 4 observations to form the POMDP shown on the right: $o_0 = \{s_0\}$, $o_1 = \{s_1, s_2\}, o_2 = \{s_3\}$, $o_3 = \{s_4\}$. This means that if the agent receives an observation o_1, the environment is either in s_1 or in s_2, and the agent has no way to determine this. The policy $\pi(p)$ shown on the right takes action a_1 with probability p in o_0, and takes a_2 with $1-p$ in o_0, and takes random uniform action in o_1. If we execute the policy $\pi(p)$ then the probability of going to o_2 or to o_3 from o_1 depends on this initial choice: we go to o_2 with probability p and to o_3 with probability $1-p$. For features, we use 1-hot encoding for the states and actions. In Appendix F we show that the optimal graph for policy $\pi(p)$ assigns probabilities $p, 1-p$ respectively to future observations o_2, o_3 at state o_1.

because, first, G^* perfectly fits data from the random policy as well, so the graph $G^r = \arg\min_G L_{\pi_r}(G)$ found from π_r would have zero loss on π_r as well. Secondly, data from a random policy contains data from any policy, with some probability, therefore, G^r has to fit data from all policies. In that case, interventions are not required to resolve spurious correlations: running a random policy is sufficient to uncover the causal structure of the environment. One of the cases which falls into this category are fully-observable deterministic environments with one-hot encoding for states as features.

In contrast, in cases where we cannot fit the data perfectly ($L_{\pi,E}(G^*) > 0$), it is possible that two policies produce different graphs, no matter the length of the history t or the method to learn the graph.

Indeed, consider the MDP shown in Figure 3. The causal graph G^p minimizing $L_{\pi(p),t}(G^p)$ depends on the policy parameter p. In Appendix F we show that for $p = 0$ the optimal prediction for observation o_1 is to always predict o_3, and for $p = 1$ the answer should be o_2. For $p \in (0, 1)$ the optimal prediction should assign the values of $p, 1-p$ for 1-hot encodings of o_2, o_3 respectively. This shows that the graph depends on the executed policy.

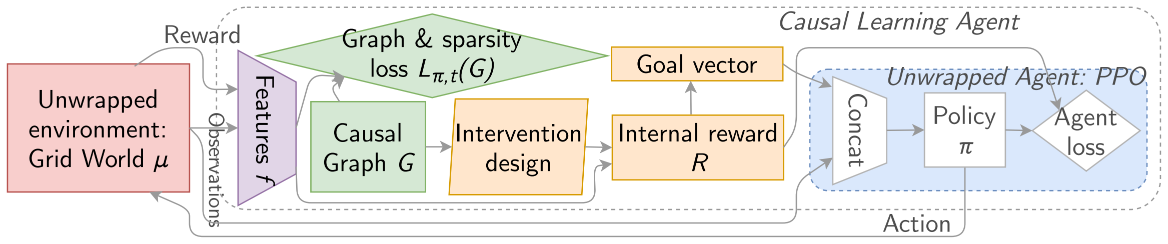


Figure 4: Learning the causal graph by actively interacting with the environment. Given a high-level set of features f_i and an environment μ, we collect data using a policy π to learn an initial causal model G from features time series. Then, we design an *intervention reward* which incentivises the agent to execute a policy which performs an intervention. The agent is trained on the intervention reward with standard reinforcement learning. The causal model is trained on the new interaction histories from the trained agent.

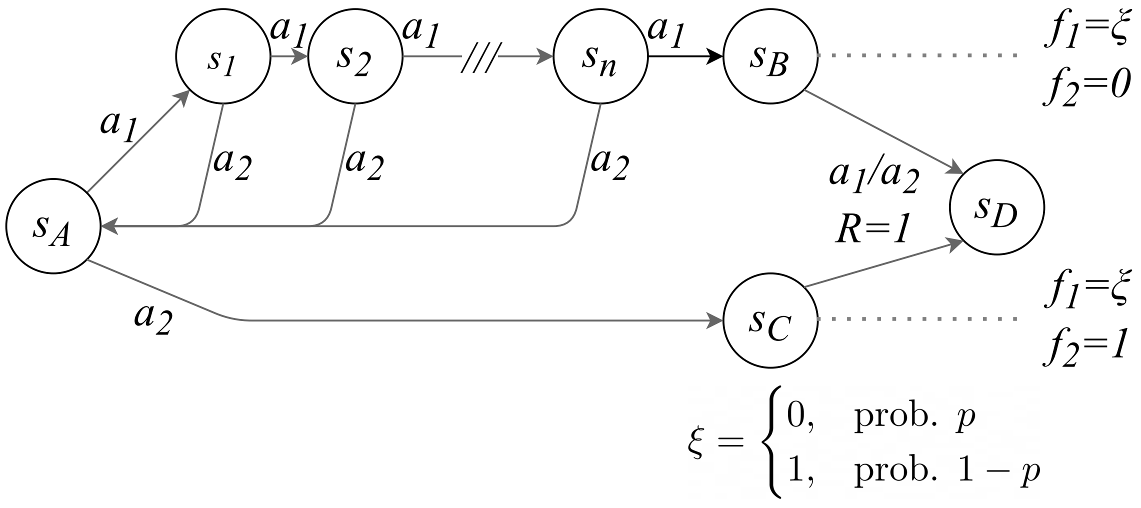


Figure 5: Environment where spurious correlations arise on a random policy but disappear in the minimax case. Specifically, we want to predict the reward R from features f_1, f_2. These features are 0 except for states s_B and s_C. A large number n leads to the random policy failing to ever visit s_n, which means that the causal graph will have much more data from s_C than from s_B. Therefore, it will learn to use f_2 to predict the reward. In contrast, if we discover that we can go to s_B by always executing action a_1, then the causal graph will favor relying on f_1, because doing so leads to a lower loss in both s_B and s_C when predicting the reward. Therefore, the minimax definition would rely on f_1. **Full description:** In our MDP, there are $n+4$ states: $s_A, s_B, s_C, s_D, s_1, ..., s_n$ with s_A being the initial state. There are two actions, a_1 and a_2. The states are visible to the policy, but only the values of f_1, f_2 are visible to the causal learner. From the initial state, we go up if we take a_1 and continue going to the right if we take a_1 until we reach s_B. If we take a_2 in any of $s_1, ..., s_n$, we return to s_A. If we take a_2 in s_A, we go to s_C. The reward is always 0 except for the transitions $s_B \to s_D$ and $s_C \to s_D$ which both generate a reward of 1 regardless of the action taken. The state s_D is terminal. There are two additional features f_1 and f_2, and they are always zero except for s_B where they take values $f_1 = \xi$ and $f_2 = 0$, and s_C where they are $f_1 = \xi$ and $f_2 = 1$. The random variable ξ is independent from all the other random variables and equals 1 with probability $1-p$ and 0 with probability p. We consider a policy which eventually arrives at s_B with probability q and at s_C with probability $1-q$. The probability q can take the value of 0 if we take an action a_2 in s_A or 1 if we always take a_1. If we take random actions in $s_1, ..., s_n$ and take a_1 in s_A, then this probability equals $\Theta(1/2^n)$. The fully random-uniform policy has $q = \Theta(1/2^n)$ as well.

Now, to understand spurious correlations, consider the MDP on Figure 5. Consider the task of predicting the reward R only from features f_1, f_2. If we rely on f_1, our model works regardless of being in s_B or s_C but there is a level of noise: the loss $L = p$. If we rely on f_2, our model works perfectly in s_C but fails in s_B. In Appendix F we show that a random uniform policy will result in relying on f_1 only for small enough $n \ll \log 1/p$. A random policy is very unlikely to reach s_B so a causal model would not learn to distinguish s_B from states which do not lead to reward. This is similar to the example with the rooster from the Introduction: in case if the agent is located in an area with roosters, it will naturally rely on the rooster to "predict" the sunrise. However, a sequence of hard-to-discover "deliberate" actions (such as moving to a far-away area) will force the model to rely on a more robust feature, such as a measurement of the relative position of the Sun with respect to Earth, even if this measurement is noisy.

In the next subsection we formulate the *minimax* causal graph. That definition would favor relying on f_1 instead of f_2, because there exists a policy (always choosing a_1) which gives a very high loss for relying on f_2. We say that using f_2 to predict the reward is a spurious correlation, since this situation is identical to the example with the rooster from the introduction.

Now, imagine that noise in feature f_1 is zero, which would happen if $p = 0$. In that case, for any policy that visits s_B at least once during data collection, the optimal model relies on f_1. Intuitively, random policy fails for that MDP with $p > 0$ simply because it does not visit one of the states often enough. Therefore, the model has a lower loss if it relies on spurious correlations (feature f_2). A way to overcome that is to develop a policy that visits s_B more often. In the next section, we introduce a way to design such policies, which we would call *interventions.*

Causal model of an environment. For all the cases, we define the problem in a minimax fashion[3]: the graph should not be disproved even by the worst policy π, at any number of collected episodes t. This is similar to the scientific method in the real world: we want to learn causal relationships that are true no matter which experiments or actions we perform.

Definition 1 (Minimax definition of the causal graph)**.** *For an environment μ, we define the causal graph G^* as the one which gives the smallest possible loss L even on the worst policy:*

$$G^* = \arg\min_G \max_\pi L_\pi(G) = \arg\min_G \max_\pi \lim_{E\to\infty} L_{\pi,E}(G) \tag{3}$$

In practice, in the equation above, we use a finite sequence of policies, instead of all policies in $\max_\pi$, and we consider a finite $E < \infty$. This can be seen as a sample estimate of the Equation 3.

As noted in the previous subsection, the Definition 1 would favor relying on a noisy feature f_1 in the example from Figure 5 achieving a constant non-zero loss in all cases, instead of relying on a more risky feature f_2 which would only work in s_C but not in s_B. In this way, the minimax definition favours "safe" solutions instead of "risky" ones.

Specifically, after executing a set of policies $\pi_1, ..., \pi_s$ and obtaining graphs $G_1, ..., G_s$, we would like to compute the policy π_{s+1} that would obtain as much information to improve G_{s+1} over previous graphs as possible. Executing the next policy π_{s+1} after π_s can be seen as doing an intervention $do(I_{s+1})$[4] in the causal model, since the policy sets nodes to specific values. In that sense, we sample novel histories from the interventional distribution $\mathbb{P}[h|do(I_{s+1})]$. Instead of specifying the policy π_{s+1} directly, we specify the reward for that policy, and we call the algorithm that designs this reward the *intervention designer.*

In the next section we show how to design interventions. The complete setup is shown in Figure 4.

3 Solution

Before, we defined a way to obtain a causal graph of a reinforcement learning environment. To deal with spurious correlations, we design the intervention reward based on the current causal graph. Next, we combine data generated from an agent trained on that reward with previously collected data to learn a better causal graph. In the next section we give concrete methods for intervention reward design. These are algorithms that determine which reward to give the agent to come up with a better causal model.

Intervention design via edges. We test if edges in the learned graph G are "real" or caused by spurious correlations. We test an edge $e = f_i \to f_j$ with a positive coefficient in a linear causal model[5], by setting $do(f_i = \max f, f_j = \min f)$ (minimal and maximal feature values). To do so, we reward the agent for setting

[3]Another approach would be to select top-k best causal models and then define the loss as how well at least one of the can predict the data.

[4]Operator $do(\cdot)$ sets (Pearl & Mackenzie, 2018; Halpern & Pearl, 2005; Pearl, 2018) the value for some nodes in the model, without changing the values of its parents, but changing the values of downstream nodes.

[5]In the non-linear case, the coefficient might depend on the current values of features. In that case, this approach will still work but the step has to be small enough to allow for such linearization.

$f_i = \max f$ and for $f_j = \min f$. The total reward is $R = f_i - f_j$ [6]. When optimizing for it, the agent does its best in executing an intervention $do(f_i = \max f, f_j = \min f)$. This reward will encourage the agent to find data for which f_i has a high value and f_j has a low value. Given that the causal model initially assumed that f_i has a positive effect on f_j, such data may disprove the causal model. To obtain the best causal graph, we select edges which the learning algorithm is uncertain about more often than those which the algorithm is certain about. We measure uncertainty by how much different the model predictions are if trained on different subsets of the history.

Intervention design via nodes. We reward the agent for setting a target node f_i to a target value x. We also reward for "keeping everything else the same". We achieve the latter either by a) penalizing for the difference d between statistics (the averages and variances) of feature distributions from previous and current policies: $R = -|f_i - x| - d$ or b) keep and down-scale the reward from the previous iteration: $R = -|f_i - x| + \gamma R_{old}$. When optimizing for this reward, the agent does its best in executing an intervention $do(f_i = x)$. We choose nodes in the graph based on the total uncertainty of adjacent edges (like in the previous technique, Edge-based intervention design, the uncertainty is computed via the variance over models trained on different subsets of the data).

Intervention design via the loss. We reward the agent for finding policies π which give high causal graph loss: $R = \sum_i \left(f_i^t - \tilde{f}_i^t\right)^2$. The reasoning behind this is that we want to find data disproving our model, like in Eq. 3. Compared to previous methods, we do not select the node or an edge explicitly. This reward design is similar to the curiosity approaches (Pathak et al., 2017)[7].

To sum up causal learning, in our approach, we simply regress the current time-step f_t on the previous one f_{t-1}: $f_t = W f_{t-1} + b$ using a linear relationship with l_1 regularization. It is trained in a (self-)supervised manner by aggregating data from different policies to approximate a solution to Eq. 3. We note that better methods (Runge et al., 2019) of learning causal graphs would still fail without interventions[8]. Other causality learning methods are compatible with our approach, since we only require a graph as an output, and we give observational data as input. Methods which discover the features end-to-end (Thomas et al., 2018; Kurutach et al., 2018; Ke et al., 2019; François-Lavet et al., 2018; Zhang et al., 2019) can be used to discover the nodes in the causal graph.

Now, all the discussed components are combined together into a causal agent, to discover the true causal graph of the environment, see Figure 4.

4 The environment

We use a simple[9] Grid-World environment. The agent needs to eat food in the environment, or the episode will end. In addition, it collects keys to open chests, with each chest giving a reward. There is a button which turns a light on and off and does not give reward. A successful policy needs to balance between staying alive (collecting food), collecting keys and chests (with keys required before opening a chest) and exploring the effect of the button. Figure 1a represents the causal model we want to discover. Appendix C contains more details about the environment.

We use the following specific environments:

(A) 5x5 grid-world with randomly placed items.

(B) (Figure 1b) a grid-world with a fixed map, where the agent must collect the key before the food.

(C) (Figure 1c) 10x10 grid-world with randomly placed items where the food is close to the key, and the chest is far away.

Since the environment A is random, the random policy is expected to uncover the correct graph, as any spurious correlation will disappear on the new layout.

In environments B, C it is easy to learn the spurious correlation in which getting a key causes an increase in health (while health is actually determined by the food collected, see Figure 6), because the agent will usually have a key when it collects the food. To uncover the correct model, the agent needs to get rid of the key by opening a chest. After that, it can collect food without having the key, and then the model will rely on the food collected to predict health.

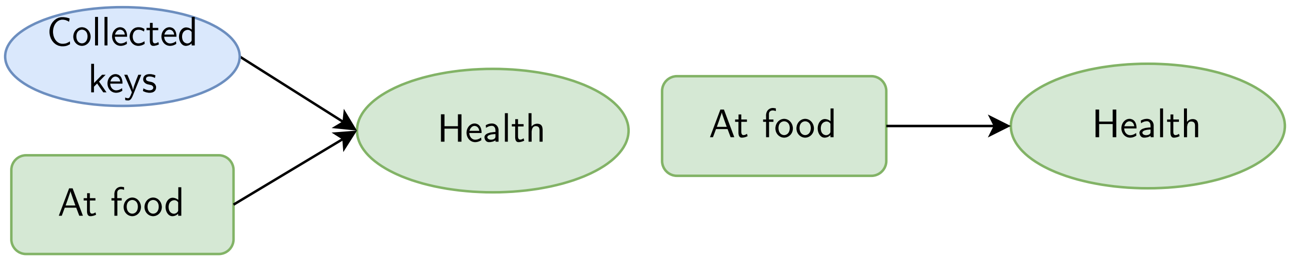
[6] In case with a negative coefficient in the model, we need $f_i + f_j$.

[7] The difference is that here, we fix the curiosity reward for many time-steps, even when a better model is available. This can be sometimes theoretically more optimal than updating the model as soon as the new data is available (Leike et al., 2016)

[8] Indeed, they have to rely on the observational data. If the data suggests a relationship between features, they have to use it, otherwise they will give a wrong answer on a case where this relationship is actually causal.

[9] While the environment is simple, a random policy or a policy trained with the reward does not obtain the true causal graph in all settings (see subsection 5.1).

We add noise to each of the environments. With some probability, food is visible at cells not containing food. Thus, we make the environment similar to the MDP from Figure 5. Indeed, there is the easy-to-discover path that leads to the wrong causal model (keys “cause” health to increase), while more exploration (getting rid of the key) yields the correct model (food causes health to increase).



(a) Wrong causal model for Environment C

(b) Correct causal model for Environment C

Figure 6: Two (parts of) models for Environments B, C (from Figure 1), in terms of nodes from Figure 1a. Left (6a): the model learned on a random policy or on the policy maximizing for the reward. The collected keys are spuriously correlated with health, because on these policies, it is likely that the agent collects the key before collecting food (collecting *food* actually results in an increase in health). Right (6b): the model learned using our method. The parent for health is correctly identified as collecting food.

The environment we choose is characteristic of the real world, as it contains spurious correlations that we need to uncover by changing the behavior.

5 Experiments

Hardcoded features. We augment the feature set with conjunctions of relevant features in order to keep the problem in the linear domain. This allowed us to keep the causal learning simple to focus on interventions. The many techniques (Ke et al., 2019; Goudet et al., 2018; Chalupka et al., 2014) for learning non-linear causal graphs that are compatible with our approach, as we only require learning from observational data.

Baselines and methods. For all methods we first use a random policy for exploration of the environment. Next, we train a PPO (Schulman et al., 2017) agent to follow the intervention reward. We compare the three proposed methods for interventions: rewarding the agent proportional to the loss of the model (Loss), disproving edges (Edge), setting nodes to values (Node). We measure if the correct graph was learned using cosine similarity between the node adjacency matrices of the currently learned graph and the ground truth.

5.1 Results

The random and environment-reward policies discover the true causal graph in environment A. Since the environment is randomly generated, there are no spurious correlations. The random and environment-reward policies extremely rarely discover the true causal graph in environments B and C. This is because the food presence feature is noisy and a random policy often collects the food before the keys because they are close together. To predict the health increase, it is best to rely on the spurious feature “food and keys > 0”. This is similar to what happens in the MDP from Figure 5 and in the rooster example from the Introduction.

The intervention methods discover the true causal graph in environments B and C more often (results for C in the appendix). This is because they also include data from the intervention policy which collects the food when the agent does not have a key. With data from the intervention policy, it is no longer optimal to rely on the spurious feature: after collecting the key and opening the chest, the key disappears, and the food is collected without the presence of keys.

The Loss intervention method outperforms Node and Edge methods on environment (B). The main problem with the Node and Edge methods is that they have to choose the correct node or edge to intervene on. Once the correct edge or node is chosen, the true graph is learned quickly. In contrast, the Loss method does not have to choose the right thing to intervene on. In simple environments hand-designed methods (edges and nodes) perform reasonably well. However, if we increase the number of features or the complexity of the environment, they stop finding good policies. We did not test the node and edge interventions in environment C extensively due to this reason.

Sampling without replacement when selecting edges gives faster convergence to the true graph versus sampling with replacement. We also found that selecting edges based on uncertainty (by training on different subsets $s = 5$) gives better results than selecting random edges. Results for selecting nodes based on uncertainty are similar to selecting randomly. For the nodes method, we found that keeping the new policy close to the old

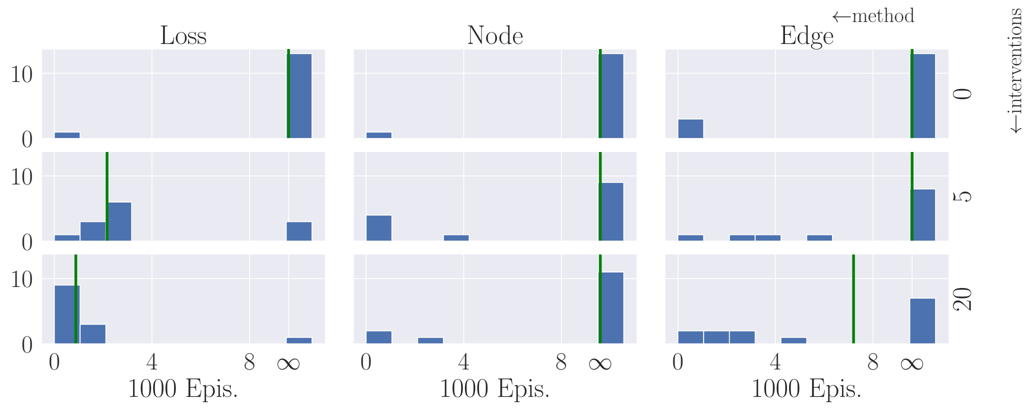


Figure 7: Experimental results on environment B for predicting the true causal graph for the health. Plots are arranged by intervention method (columns: Loss, Node, Edge) and by the number of interventions performed (rows: 0, 5, 20). Each plot shows the histogram for the number of episodes (in 1000s) it takes to find the true causal graph G^*. The horizontal axis shows the number of episodes, and the vertical axis represents the number of runs (out of 10) which have converged to the true graph G^*. ∞ means the graph was not found during training. Green line represents the median number of episodes it takes to find the correct graph. 0 interventions corresponds to training with the original environment reward. The random policy is evaluated in a separate experiment with spurious correlations as a result. The overall trend shows that the median number of episodes before uncovering the correct graph decreases for Loss and Edge methods, as the number of interventions increases.

one by using the reward from the environment works. However keeping the feature statistics the same does not work because the agent learns a new way to achieve the same statistics.

6 Conclusion

We design a method to learn the causal model of the environment by performing interventions, which helps prevent learning spurious correlations. This shows the potential of RL to improve causal graph learning and compares techniques to accomplish this. We state the problem of learning a causal graph in a RL setting, so other work can build off of ours.

7 Future Work

We plan to combine our graph learning with one of the approaches for learning the features (Kurutach et al., 2018; Ke et al., 2019; François-Lavet et al., 2018; Zhang et al., 2019) and train the entire network end-to-end. The sparsity loss will help the features be disentangled because it will minimize dependencies between them (Thomas et al., 2018).

To make our method more general, we plan to use one of the advanced non-linear causality learners (Ke et al., 2019). Some of them are differentiable, which would allow to backpropagate from the graph to the features.

Finally, we can utilize the high-level graph as a hierarchical RL controller (Nachum et al., 2018). Specifically, we can run a traversal algorithm on the causal graph to find chains of nodes that lead to high reward. Then, we can reward the agent for activating the nodes in the correct sequence. This might increase the robustness to distributional shift, as we will rely on the correct features for acting.

8 Acknowledgements

We thank Ivan Vendrov, Rohin Shah, Jacob Hilton, Honglak Lee, Lars Buesing, Prashan Madumal for helpful discussions and feedback on drafts of this paper. We thank Stuart Russell for pointing us to similar projects. Thanks to Anastasia Koloskova, Olesya Altunina, David Lindner, Todor Markov, Denis Drescher, Dmitry Mironov, Igor Molybog, Valentin Hartmann and Bryce Woodworth for providing feedback on the draft and videos and to El-Mahdi El-Mhamdi for introducing us to the consciousness prior. Thanks to David Krueger for suggesting an improved version of the causal learning objective function.

References

Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, et al. Solving rubik's cube with a robot hand. *arXiv preprint arXiv:1910.07113*, 2019.

Yoshua Bengio. The consciousness prior. *arXiv preprint arXiv:1709.08568*, 2017.

Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Dębiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, et al. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*, 2019.

Krzysztof Chalupka, Pietro Perona, and Frederick Eberhardt. Visual causal feature learning. *arXiv preprint arXiv:1412.2309*, 2014.

Pim de Haan, Dinesh Jayaraman, and Sergey Levine. Causal confusion in imitation learning. In *Advances in Neural Information Processing Systems*, pp. 11693–11704, 2019.

Tom Everitt, Pedro A Ortega, Elizabeth Barnes, and Shane Legg. Understanding agent incentives using causal influence diagrams, part i: single action settings. *arXiv preprint arXiv:1902.09980*, 2019.

Vincent François-Lavet, Yoshua Bengio, Doina Precup, and Joelle Pineau. Combined Reinforcement Learning via Abstract Representations. sep 2018. URL `http://arxiv.org/abs/1809.04506`.

Adam Gleave, Michael Dennis, Cody Wild, Neel Kant, Sergey Levine, and Stuart Russell. Adversarial policies: Attacking deep reinforcement learning. *arXiv preprint arXiv:1905.10615*, 2019.

Olivier Goudet, Diviyan Kalainathan, Philippe Caillou, Isabelle Guyon, David Lopez-Paz, and Michele Sebag. Learning functional causal models with generative neural networks. In *Explainable and Interpretable Models in Computer Vision and Machine Learning*, pp. 39–80. Springer, 2018.

Clive WJ Granger. Investigating causal relations by econometric models and cross-spectral methods. *Econometrica: journal of the Econometric Society*, pp. 424–438, 1969.

David Ha and Jürgen Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2018.

Joseph Y Halpern and Judea Pearl. Causes and explanations: A structural-model approach. part i: Causes. *The British journal for the philosophy of science*, 56(4):843–887, 2005.

Alex Irpan. Deep reinforcement learning doesn't work yet. `https://www.alexirpan.com/2018/02/14/rl-hard.html`, 2018.

Khurram Javed, Martha White, and Yoshua Bengio. Learning causal models online. *arXiv preprint arXiv:2006.07461*, 2020.

Nan Rosemary Ke, Olexa Bilaniuk, Anirudh Goyal, Stefan Bauer, Hugo Larochelle, Chris Pal, and Yoshua Bengio. Learning Neural Causal Models from Unknown Interventions. oct 2019. URL `http://arxiv.org/abs/1910.01075`.

Andrey Kurenkov. Reinforcement learning's foundational flaw. *The Gradient*, 2018.

Thanard Kurutach, Aviv Tamar, Ge Yang, Stuart Russell, and Pieter Abbeel. Learning Plannable Representations with Causal InfoGAN. jul 2018. URL `http://arxiv.org/abs/1807.09341`.

Jan Leike, Tor Lattimore, Laurent Orseau, and Marcus Hutter. Thompson sampling is asymptotically optimal in general environments. *arXiv preprint arXiv:1602.07905*, 2016.

Prashan Madumal, Tim Miller, Liz Sonenberg, and Frank Vetere. Explainable Reinforcement Learning Through a Causal Lens. may 2019. URL `https://arxiv.org/abs/1905.10958`.

Kenneth Marino, Rob Fergus, and Arthur Szlam. Toward a Scientist Agent : Learning to Verify Hypotheses. 2019.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Ofir Nachum, Shixiang Shane Gu, Honglak Lee, and Sergey Levine. Data-efficient hierarchical reinforcement learning. In *Advances in Neural Information Processing Systems*, pp. 3303–3313, 2018.

Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 16–17, 2017.

Judea Pearl. *Causality*. Cambridge university press, 2009.

Judea Pearl. Theoretical impediments to machine learning with seven sparks from the causal revolution. *arXiv preprint arXiv:1801.04016*, 2018.

Judea Pearl and Dana Mackenzie. *The book of why: the new science of cause and effect*. Basic Books, 2018.

Jakob Runge, Sebastian Bathiany, Erik Bollt, Gustau Camps-Valls, Dim Coumou, Ethan Deyle, Clark Glymour, Marlene Kretschmer, Miguel D Mahecha, Jordi Muñoz-Marí, et al. Inferring causation from time series in earth system sciences. *Nature communications*, 10(1):1–13, 2019.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Valentin Thomas, Emmanuel Bengio, William Fedus, Jules Pondard, Philippe Beaudoin, Hugo Larochelle, Joelle Pineau, Doina Precup, and Yoshua Bengio. Disentangling the independently controllable factors of variation by interacting with the world. *arXiv preprint arXiv:1802.09484*, 2018.

Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019.

Amy Zhang, Zachary C. Lipton, Luis Pineda, Kamyar Azizzadenesheli, Anima Anandkumar, Laurent Itti, Joelle Pineau, and Tommaso Furlanello. Learning Causal State Representations of Partially Observable Environments. (ii):1–16, 2019. URL `http://arxiv.org/abs/1906.10437`.

A Relevant work

Our method to perform an intervention on an edge is similar to the method used in Marino et al. (2019) to test hypotheses. Compared to that approach, we are interested in the true causal graph of the environment rather than in testing specific hypotheses. Interventions to learn the true graph can be seen in (de Haan et al., 2019). Our approach is focused on learning the correct graph rather than acting well. We extend the Action-Influence model (Marino et al., 2019; Everitt et al., 2019) to understand the environment. Compared to (Madumal et al., 2019), we learn the graph rather than design it by hand. The idea to reward the agent for the loss of the causal graph is taken from (Pathak et al., 2017). However, here we are interested in a very low-dimensional causal graph rather than in a black-box model of the environment. Compared to standard model-based techniques (Ha & Schmidhuber, 2018), our approach filters spurious correlations in the model.

A.1 Relevant work after the first pre-print release

Compared to (Javed et al., 2020), we are learning the causal graph of practical RL environments, we define the minimax graph and have a comparison of various intervention techniques.

B Hyperparameter selection

Resources and parameters. Parameters were chosen with a hyperparameter search on the task of solving the environment using PPO (Schulman et al., 2017). We run the total of 8000 episodes for B and 50000 episodes for C. We vary the number of epochs to train the causal graph in 500-10000, number of interventions 0-50, number of training calls 5-100, intervention method Loss, Edge, maximal number of episodes in the buffer 10-5000, method to select the edge Constant, Weighted and Random. We learn the graph on evaluation data without noise, and update the reward for the trainer.

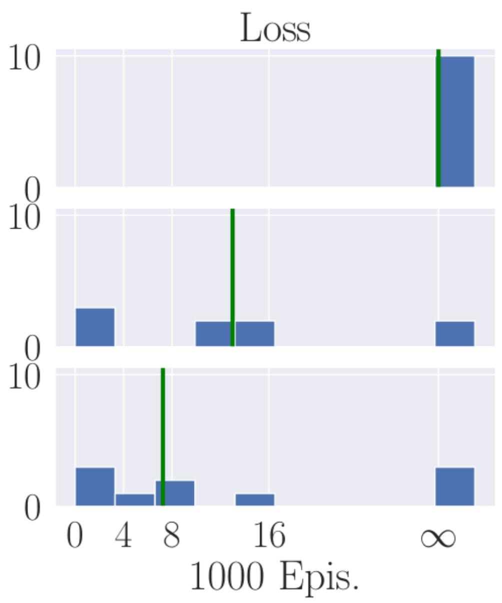


Figure 8: Experimental results on environment C for predicting health, reward, keys and lamp. The description matches that of Figure 7.

C The environment

We implement the environment using pycolab. All updates are delayed 1 time-step to give causal information.
An open-source re-implementation can be found in github.com/sergeivolodin/causality-disentanglement-rl.

D Experiments

Figure 8 shows the results for environment C. Without interventions, the correct graph is never uncovered. In contrast, with interventions, the correct graph is learned, the more interventions the better.

E Realizable case

Proposition 1. *For an environment μ and features f, learner L which can always converge to a global minimum for any data distribution, if a true causal graph is G^* (s.t. $L_\pi(G^*) = 0$ for all π and $L > 0$ for other G), a random policy π_r gives the true graph:* $\min_G \max_t L_{\pi_r,E}(G) = \min_G \max_\pi \max_E L_{\pi,E}(G^*)$

Note that in our experiments, we use a linear model which complies with this property. Since a linear model is always convex, its optimization procedure always finds a global minimum. In contrast, we expect that more complex causality learners would benefit significantly from doing directed interventions rather than doing random exploration: Proposition 1 no longer applies to them.

Intuitively, in the realizable case, more data is always better. The proof is based on two ideas: first, for a policy π^* giving the true G^*, a random policy π_r will take same actions as π^* with some probability: $\mathbb{P}[\pi_r = \pi^*] > 0$. Thus, data from π^* will be in the dataset. Next, since G^* fits any policy, the learner will find a graph G s.t. $L_{\pi_r,E}(G) = 0$. By linearity of L, loss on π_r equals a non-negative combination of losses over policies π_r equals to, including one for π^*. Now, since the non-negative combination is 0, one particular term $L_{\pi^*,E}(G) = 0$ as well, and $G = G^*$. The full proof is below.

Proof. Consider data (histories) h coming from the agent-environment interaction. We write $h \sim \pi$ meaning that h is a history obtained from interaction between a policy π and our fixed environment μ which we omit in this notation. If $E(h, G)$ corresponds to the error of the model given by a causal graph G on the history h, we have by definition $L_{\pi,E}(G) = \frac{1}{E}\sum\limits_{e=1}^{E} \mathbb{E}(h_e, G)$. If we take $\max_e$, there is a limit $L_{\pi,E}(G) \to \mathbb{E}_{h\sim\pi}\mathbb{E}(h, G)$ as $e \to \infty$: if we add more data to the learner, it is the same as training on "infinite data" in the sense of the distribution.

By definition of G^*, we have $\forall G\ \max_\pi \max_E L_{\pi,E}(G) \geq \max_\pi \max_E L_{\pi,E}(G^*)$. We define $L_\pi(G) = \max_E L_{\pi,E}(G)$ and $L(G) = \max_\pi L_\pi(G)$, then we have $L(G) \geq L(G^*)$, which means that G^* is the global minimum of $L(\cdot)$ whose numerical value we write as $L_{\text{min}} = L(G^*)$. By the definition of $L(G^*) = \max_\pi L_\pi(G^*) =$

$L_{\min} \in \mathbb{R}$. Also by the theorem statement, G^* achieves the minimal value of L given any policy: we have $L_\pi(G^*) = L_{\min}$ for all π. This means that G^* is also a global minimum for L_π for each π.

Next, consider a random policy π_r which is a random variable over the space of all deterministic policies. We write some random outcomes $\omega \in \Omega$ meaning the probability space of all deterministic policies. The interpretation is that by the "inverse" of the Principle of Deferred Decisions (we pretend that we made all the random choices in advance rather than making them as they are required), there are some factors ω that influence the agent-environment interaction. Once they are chosen, the policy π_r is just some deterministic policy for that ω. Then, we can write $L_{\pi_r}(G) \equiv \mathbb{E}_{h\sim\pi_r} E(h, G) = \mathbb{E}_{\omega\in\Omega}\mathbb{E}_{h\sim\pi_r} E(h, G)$. Now, for each outcome $\omega \in \Omega$, π_r is a deterministic policy, and then, $\mathbb{E}_{\omega\in\Omega}\mathbb{E}_{h\sim\pi_r} E(h, G) \equiv \mathbb{E}_{\omega\in\Omega}\mathbb{E}_{h\sim\pi_\omega} E(h, G)$. Thus, we can write $L_{\pi_r}(G) = \mathbb{E}_{\omega\in\Omega}\mathbb{E}_{h\sim\pi_\omega} E(h, G) \equiv \mathbb{E}_{\omega\in\Omega} L_{\pi_\omega}(G)$

So now, $L_{\pi_r}(G) = \mathbb{E}_{\omega\in\Omega} L_{\pi_\omega}(G)$. Now, since G^* is a global minimum of L_π for any π, $L_{\pi_r}(G) \geq \mathbb{E}_{\omega\in\Omega} L_{\pi_\omega}(G^*) = L_{\pi_r}(G^*) = L_{\min}$.

Now, consider $L_{\pi_r}(G)$ from another perspective. This corresponds to a model G trained on data from π_r. If we use the fact that the learner always finds the global minimum, we know that $L_{\pi_r}(G) \leq L_{\pi_r}(G^*)$: we know that on G^* it has a certain loss. Therefore, if it has output G, the loss should be less.

Now, we have two inequalities which we merge into one equality: $L_{\pi_r}(G^*) \geq L_{\pi_r}(G) \geq L_{\pi_r}(G^*)$. Thus, we see that $L_{\pi_r}(G) = L_{\pi_r}(G^*) = L_{\min}$.

Finally, the last equality means that $L_{\pi_r}(G)$ attains the minimal possible value of L, thus, G also delivers the global minimum of $L(\cdot)$ and $L(G) = L(G^*)$ □

F MDPs from Figures

MDP from Figure 3

1. If $p = 0$, we always choose to go to s_2 from s_0. Therefore, we end up in s_4 after that, which corresponds to o_3, analogously $p = 1$ corresponds to going to s_1 and then to s_3 which corresponds to o_2

2. If we set $p \in (0, 1)$ and write the loss of a linear model with coefficients w_2, w_3 for the o_2 1-hot component and the o_3 1-hot component respectively, the loss is $L = p \cdot \left[(1 - w_2)^2 + w_3^2\right] + (1 - p) \cdot \left[w_2^2 + (1 - w_3)^2\right]$. Minimizing this expression gives $w_2 = p$, $w_3 = 1 - p$, $L = 2p - 2p^2$.

3. If we consider the minimax case, we need to select a graph which results in the smallest loss in the worst case. To do that, we first maximize the previous expression for L over p, and then minimize it over w_2, w_3, which gives $w_2 = w_3 = 1/2$, $L = 1/2$ and does not depend on p.

MDP from Figure 5

1. The linear reward-predicting model must have a bias of 0 so that it predicts zero rewards in states $s_A, s_1, ..., s_n$ (when n is large enough, this becomes significant).

2. If we take a_2 at the beginning, the loss of a linear model predicting the reward $s_C \to s_D$ would be $(1 - w_1\xi - w_2 \cdot 1)^2$. The expectation of this becomes $(1 - p)(1 - w_1 - w_2)^2 + p(1 - w_2)^2$. Minimizing this over w_1, w_2 gives $w_1 = 0$ and $w_2 = 1$: we rely on the feature f_2, and the best loss is $L = 0$

3. If we always take a_1, the loss for predicting the reward at $s_B \to s_D$ would be $(1 - w_1\xi)^2$. Its expectation is $p + (1 - p)(1 - w_1)^2$, the best loss is $L = p$ and $w_1 = 1$, $w_2 = 0$ (if noise parameter $p < 1$). We rely on f_1, even if it is noisy.

4. Now, if we take a policy which arrives to s_B with probability q, the loss becomes a linear combination of the two losses with coefficients q, $1-q$, the weights depend on the coefficients p, q. Specifically, consider a policy arriving at s_B with probability q. For a random uniform policy, $q = \Theta(1/2^n)$ (up to a constant). Now, the loss becomes $L = q \cdot \left[p + (1 - p)(1 - w_1)^2\right] + (1 - q) \cdot \left[(1 - p)(1 - w_1 - w_2)^2 + p(1 - w_2)^2\right]$. This function attains a minimum at a point with $w_1 = q/(p + q - pq)$. If $p \leq q\alpha$, $w_1 = q/(q\alpha + q - q^2\alpha) \geq 1/(1 + \alpha)$. For $\alpha = 1/100$, we get that $w_1 \geq 1 - 1/100 \approx 1$. This shows that for $p \ll q$, $w_1 \approx 1$. For a random uniform policy this translates to $p \ll 1/2^n$ which means that $n \ll -\log p$

5. If we consider the minimax case, we need to select a graph which results in the smallest loss in the worst case, which would mean relying on f_1 and obtaining a worst-case loss of p (worst-case $q = 0$) instead of relying on f_2 and obtaining a worst-case of loss 1 (worst case $q = 1$).