Deconfounding Reinforcement Learning in Observational Settings

Chaochao Lu   CL641@CAM.AC.UK
Department of Engineering
University of Cambridge
Cambridge, United Kingdom

Department of Empirical Inference
Max Planck Institute for Intelligent Systems
Tübingen, Germany

Bernhard Schölkopf   BS@TUEBINGEN.MPG.DE
Department of Empirical Inference
Max Planck Institute for Intelligent Systems
Tübingen, Germany

José Miguel Hernández-Lobato   JMH233@CAM.AC.UK
Department of Engineering
University of Cambridge
Cambridge, United Kingdom

Microsoft Research
Cambridge, United Kingdom

Alan Turing Institute
London, United Kingdom

Abstract

We propose a general formulation for addressing reinforcement learning (RL) problems in settings with observational data. That is, we consider the problem of learning good policies solely from historical data in which unobserved factors (confounders) affect both observed actions and rewards. Our formulation allows us to extend a representative RL algorithm, the Actor-Critic method, to its deconfounding variant, with the methodology for this extension being easily applied to other RL algorithms. In addition to this, we develop a new benchmark for evaluating deconfounding RL algorithms by modifying the OpenAI Gym environments and the MNIST dataset. Using this benchmark, we demonstrate that the proposed algorithms are superior to traditional RL methods in confounded environments with observational data. To the best of our knowledge, this is the first time that confounders are taken into consideration for addressing full RL problems with observational data. Code is available at https://github.com/CausalRL/DRL.

1. Introduction

In recent years, much attention has been devoted to the development of reinforcement learning (RL) algorithms with the goal of improving treatment policies in healthcare (Gottesman et al., 2018). RL algorithms have been proposed to infer better decision-making strategies

for mechanical ventilation (Prasad et al., 2017), sepsis management (Raghu et al., 2017a,b), and treatment of schizophrenia (Shortreed et al., 2011). In healthcare, a common practice is to focus on the observational setting in which we learn policies solely from historical data produced by real environments, instead of learning policies by actively taking actions in the traditional RL setting. The reason for this is that we do not wish to experiment with patients' lives without evidence that the proposed treatment strategy is better than the current practice (Gottesman et al., 2018).[1] As pointed out by Raghu et al. (2017a), RL also has advantages over other machine learning algorithms even in the observational setting, especially in two situations: when the optimal treatment strategy is unclear in medical literature (Marik, 2015), and when training examples do not represent optimal behavior.

Another approach that has been explored and used is causal inference (Pearl, 2009). Causal inference is a fundamental notion in science and plays an increasingly important role in healthcare and medicine (Liu et al.; Soleimani et al., 2017; Schulam and Saria, 2017; Alaa et al., 2017; Alaa and van der Schaar, 2018; Atan et al., 2016). Especially in the observational setting, causal inference offers a powerful tool to deal with confounding that occurs when a hidden variable influences both the treatment and the outcome (Pearl and Mackenzie, 2018). It is widely acknowledged that confounding is the most crucial aspect of inferring the effect of the treatment on the outcome from observational data (Louizos et al., 2017), because not taking confounding into account might lead to choosing a wrong treatment. However, most approaches in causal inference to tackling confounding are restricted to fixed treatments which do not vary over time (Louizos et al., 2017). When treatments are allowed to vary over time, the space of combinational treatment strategies increases exponentially (Peters et al., 2017; Hernán and Robins, 2018). In this case, it is still unclear how to apply causal inference to dealing with confounding in such sequential data where treatments changes over time.

On the basis of the discussion above, in this paper we attempt to combine advantages of RL and causal inference to cope with an important family of RL problems in the observational setting, that is, learning good policies solely from the historical data produced by real environments with *confounding bias*. This type of problem will become increasingly common in future RL research with the burgeoning development in healthcare and medicine. To the best of our knowledge, however, little work has been done in this promising area of integrating RL with causal inference (Bareinboim et al., 2015; Forney et al., 2017; Buesing et al., 2019). In contrast, confounders have been extensively studied in epidemiology, sociology, and economics. Take for example the widespread kidney stones in which the size of the kidney stone is a confounding factor affecting both the treatment and the recovery (Peters et al., 2017; Pearl, 2009). Correcting for the confounding effect of the size of the kidney stone is crucial for determining how to choose an effective treatment. Similarly, in RL, if unobserved potential confounders exist, they affect both actions and rewards as an agent interacts with environments, and eventually influence the policy to be optimized.

Let us for a moment stick with the example of the kidney stones and assume that physicians need to take a series of steps to treat this disease. We further assume that, during the course of treatment, physicians have natural predilections of the treatment choices as a function of the stone size. That is, it is more likely for them to choose some treatment when patients have large stones and to choose another treatment when patients have small stones.

---

1. Another example is in finance where, due to costs in terms of time and money, it is often impractical to evaluate trading strategies by actually buying and selling stocks in the market.

In the observational setting, given only such a set of historical records about physicians' treatments and outcomes on patients, it is extremely challenging or even impossible to learn an optimal treatment policy due to the existence of the confounder (i.e., the size of kidney stones).

To this end, we present a general formulation for addressing RL problems with observational data, namely *deconfounding reinforcement learning* (DRL). More specifically, given several common confounding assumptions, we first estimate a latent-variable model from observational data. Under some conditions for identification, through the latent-variable model we can simultaneously discover the latent confounders and infer how they affect actions and reward. Then the confounders in the model can be adjusted for, using the causal language developed by Pearl (2009), and finally we optimize the policy based on the resulting deconfounding model.

On the basis of the proposed formulation, we extend one popular RL algorithm, the Actor-Critic method, to its corresponding deconfounding variant. Note that our procedure for obtaining this deconfounding variant can be easily applied to other RL algorithms. Due to lack of datasets in this respect, we revise the classic control toolkit in OpenAI Gym (Brockman et al., 2016), making it a benchmark for comparison of DRL algorithms. In addition, we also devise a confounding version of the MNIST dataset (LeCun et al., 1998) to verify the performance of our causal model. Finally, we conduct extensive experiments to demonstrate the superiority of the proposed formulation compared to traditional RL algorithms in confounded environments with observational data.

To sum up, our contributions in this paper are as follows:

1. We propose a general formulation for addressing RL problems in confounded environments with observational data, namely *deconfounding reinforcement learning* (DRL);

2. We present the deconfounding variant of Actor-Critic methods, obtained through a methodology that can be easily applied to other RL methods;

3. We develop a benchmark for DRL by revising the toolkit for classic control in OpenAI Gym (Brockman et al., 2016) and by devising a version of the MNIST dataset with confounders (LeCun et al., 1998);

4. We perform a comprehensive comparison of our DRL algorithm with its vanilla version, showing that the proposed approach has an advantage in confounded environments.

5. To the best of our knowledge, this is the first attempt to build a bridge between confounding and the full RL problem. This is one of few research papers aiming at understanding the connections between causal inference and the full RL setting.

2. Background

In this section, we briefly review *confounding* in causal inference. We recommend Pearl's excellent monograph for further reading (Pearl, 2009).
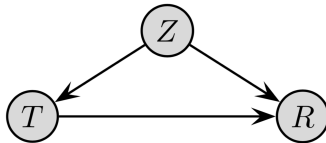
Figure 1: Causal diagram for kidney stones.

2.1 Simpson's Paradox

Let us begin with an example of one of the most famous paradoxes in statistics: Simpson's Paradox. Consider the previously mentioned kidney stones (Peters et al., 2017). We collect electronic patient records to investigate the effectiveness of two treatments against kidney stones. Although the overall probability of recovery is higher for patients who took treatment b, treatment a performs better than treatment b both on patients with small kidney stones and with large kidney stones. More precisely, we have

$$p(R = 1|T = b) > p(R = 1|T = a); \qquad \text{but}$$
$$p(R = 1|T = b, Z = l) < p(R = 1|T = a, Z = l),$$
$$p(R = 1|T = b, Z = s) < p(R = 1|T = a, Z = s); \tag{1}$$

where Z is the size (large or small) of the stone, T the treatment, and R the recovery (all binary). How do we cope with this change in conclusions? Which treatment do you prefer if you had kidney stones? Does treatment b cause recovery? As described in the following section, the answers to these questions depend on the causal relationship between treatment, recovery, and the size of the kidney stone.

2.2 Confounding

An intuitive explanation for this kidney stone example of Simpson's paradox is that larger stones are more severe than small stones and are much more likely to be treated with treatment a, resulting in that treatment a looks worse than treatment b overall. We assume that Figure 1 depicts the true underlying causal diagram of the kidney stone example. Confounding occurs because the size of kidney stones influences both treatment and recovery. The size of the kidney stones is called a confounder. The term "confounding" originally meant "mixing" in English, which describes that the true causal effect $T \to R$ is "mixed" with the spurious correlation between T and R induced by the fork $T \leftarrow Z \to R$ (Pearl and Mackenzie, 2018). In other words, we will not be able to disentangle the true effect of T on R from the spurious effect if we do not have data on Z. If we have measurements of Z or can indirectly estimate Z, it is easy to deconfound the true and spurious effects. To this end, we can adjust for Z by averaging the effect of T on R in each subgroup of Z (i.e., different size groups in the case of kidney stones).

2.3 Do-Operator and Adjustment Criterion

From the viewpoint of causal inference, we can use the language of intervention, namely the *do-operator*, to describe when confounding happens. In fact, in the example of the kidney

stones, what we are interested in is how these two treatments compare when we force all patients to take treatment a or b, rather than which treatment has a higher recovery rate given only the observational patient records. Mathematically, we focus on the true effect $p(R = 1|do(T = a))$ (the *intervention distribution* obtained when patients are forced to take treatment a) instead of the conditional $p(R = 1|T = a)$ (the *observational distribution* obtained when patients are observed to take treatment a). Therefore, as described previously, confounding can be naturally described as the discrepancy between $p(R|T)$ and $p(R|do(T))$.

The do-operator can be executed in two common ways: by Randomized Controlled Trials (RCTs) (Fisher, 1935) and by adjustment formulas (i.e., Back-door criterion and Front-door criterion) (Pearl, 2009). RCTs are the gold standard but sometimes limited by practical considerations (e.g., safety, laws, ethics, physically infeasibility, etc.). The Back-door and Front-door criteria require knowledge of the causal diagram, which typically means that causal assumptions are thus provided in advance. According to the Back-door criterion, in the kidney stone example, we can immediately obtain

$$p(R = 1|do(T = a)) = \sum_{z \in \{l,s\}} p(R = 1|T = a, Z = z)p(Z = z). \tag{2}$$

Beyond the two adjustment formulas, the *do*-calculus provides a syntactic method of deriving claims about interventions (Pearl, 2009). It consists of three rules which can be repeatedly applied to simplify the expression for an interventional distribution. With the do-calculus we can calculate intervention probabilities from observation probabilities.

2.4 Proxy Variables for Confounding

If confounders can be measured, then they can be adjusted for through the methods discussed in Section 2.3. However, in most cases where confounders are hidden or unmeasured, without further assumptions, it is impossible to estimate the effect of the intervention on the outcome. A common practice is then to leverage observed proxy variables that contain information about the unobserved confounders (Angrist and Pischke, 2008; Maddala and Lahiri, 1992; Montgomery et al., 2000; Schölkopf et al., 2016). However, using proxy variables to correctly recover causal effects requires strict mathematical assumptions (Louizos et al., 2017; Edwards et al., 2015; Kuroki and Pearl, 2014; Miao et al., 2018; Pearl, 2012; Wooldridge, 2009) and, in practice, we do not know whether or not the available data meets those assumptions. Hence, we decide to follow Louizos et al. (2017) and, instead of using proxy variables, we estimate a latent-variable model in which we simultaneously discover the latent confounders and infer how they affect treatment and outcome.

3. Deconfounding Reinforcement Learning

In this section, we will formally introduce *deconfounding reinforcement learning* (DRL). Generally speaking, DRL consists of two steps: learning a deconfounding model shown in Figure 2 and optimizing a policy based on the learned deconfounding model. The main idea in step 1 is to simultaneously discover hidden confounders and infer causal effects through the estimation of a latent-variable model. More specifically, we first discuss the time-independent confounding assumption in Section 3.1 and then, based on this assumption, we formalize the
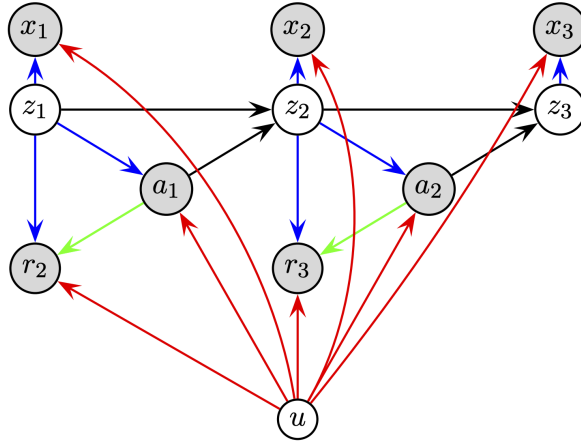
Figure 2: The model for deconfounding reinforcement learning. Grey nodes denote observed variables and white nodes represent unobserved variables. Red and blue arrows emphasize the observed variables affected by u and by z_t, respectively. The causal effects of interest are colored in green.

deconfounding model in Section 3.2. Section 3.3 talks about the problem of identification in our model, which is a central issue in causal inference. After that, we present details about how to learn the proposed model via variational inference in Section 3.4. Step 2 is provided in Section 3.5 where we describe how to design the deconfounding actor-critic method. The proposed approach is straightforward to apply to other RL algorithms in the same manner. Finally, the training procedure for our deconfounding actor-critic method is presented in Section 3.6.

3.1 Causal Assumptions

Without loss of generality, we assume that there exists a common confounder in the model, denoted by u in Figure 2, which is time-independent across a number of episodes. This assumption is sufficiently general to apply to various RL tasks across domains. For example, in personalized medicine, socio-economic status can be a confounder affecting both the medication strategy a patient has access to and the patient's general health (Louizos et al., 2017). In this case, socio-economic status is time-independent for each patient during the course of treatment. Another example is in agriculture, where soil fertility may serve as a time-independent confounder affecting both the application of fertilizer and the yield of each plot of land (Pearl and Mackenzie, 2018). Finally, in the stock market example, government policy may also act as a time-independent confounder.

3.2 The Model

Given our causal assumptions, we first fit a generative model to a sequence of observational data: observations, actions, and rewards, where actions and rewards are confounded by one or several unknown factors, as shown in Figure 2. Formally, let $\vec{x} = (x_1, \ldots, x_T)$,

$\vec{a} = (a_1, \ldots, a_{T-1})$, $\vec{r} = (r_2, \ldots, r_{T+1})$, $\vec{z} = (z_1, \ldots, z_T)$ be the sequences of observations, actions, rewards, and corresponding latent states, respectively. As mentioned previously, the confounder is denoted by u, and it is worth noting that here u may stand for more than one confounder in which multiple confounders are seen as a whole represented by u (i.e., the confounder u can be a vector). We assume that $x_t \in \mathbb{R}^{D_x}$, $a_t \in \mathbb{R}^{D_a}$, $r_t \in \mathbb{R}^{D_r}$, $z_t \in \mathbb{R}^{D_z}$, and $u \in \mathbb{R}^{D_u}$, where $D_z \ll D_x$. The generative model for DRL is then given by

$$p(z_t) = \prod_{j=1}^{D_z} \mathcal{N}(z_{tj}|0,1); \qquad p(u) = \prod_{j=1}^{D_u} \mathcal{N}(u_j|0,1);$$

$$p(x_t|z_t, u) = \mathcal{N}\left(x_t|\hat{\mu}_t^x, \hat{\sigma}_t^{x\,2}\right); \quad \hat{\mu}_t^x = f_1(z_t, u), \quad \hat{\sigma}_t^{x\,2} = f_2(z_t, u); \tag{3}$$

$$p(a_t|z_t, u) = \mathcal{N}\left(a_t|\hat{\mu}_t^a, \hat{\sigma}_t^{a\,2}\right); \quad \hat{\mu}_t^a = f_3(z_t, u), \quad \hat{\sigma}_t^{a\,2} = f_4(z_t, u); \tag{4}$$

$$p(r_{t+1}|z_t, a_t, u) = \mathcal{N}\left(r_{t+1}|\hat{\mu}_t^r, \hat{\sigma}_t^{r\,2}\right); \quad \hat{\mu}_t^r = f_5(z_t, a_t, u), \quad \hat{\sigma}_t^{r\,2} = f_6(z_t, a_t, u); \tag{5}$$

$$p(z_t|z_{t-1}, a_{t-1}) = \mathcal{N}\left(z_t|\hat{\mu}_t^z, \hat{\sigma}_t^{z\,2}\right); \quad \hat{\mu}_t^z = f_7(z_{t-1}, a_{t-1}), \quad \hat{\sigma}_t^{z\,2} = f_8(z_{t-1}, a_{t-1}); \tag{6}$$

where we have parametrized each probability distribution as a Gaussian with mean and variance given by nonlinear functions f_k represented by neural networks with parameters θ_k for $k = 1, \ldots, 8$. Note that, in our case, Equation (4) is not necessary when learning the model, because the data used in our experiments are generated from a policy depending only on the confounder (see Section 4.1), that is, a_t does not depend on z_t in our data. In addition, in this case z_t is not viewed as a confounder of a_t and r_{t+1}, for z_t does not influence a_t. This also provides one reason why we do not need to adjust for z_t. However, in some other cases such as when applied to medical data, the strategies of treatment from physicians definitely contain valuable information about z_t and a_t (i.e., z_t does influence a_t), and therefore the arrow between them is necessary when learning the model.

3.3 Identification of Causal Effect

The key component of our method that allows us to address problems with confounders is the computation of the reward according to the model from Figure 2. To be more precise, assuming that an agent standing at state z_t performs action $a_t = \mathfrak{a}$, we do not use the conditional $p(r_t|z_t, a_t = \mathfrak{a})$ as predictor for the reward as traditional RL methods do. Instead, we use the do-operator described in Section 2.3 to compute the reward[2]:

$$p(r_t|z_t, \mathrm{do}(a_t = \mathfrak{a})) = \int_u p(r_t|z_t, \mathrm{do}(a_t = \mathfrak{a}), u)p(u|z_t, \mathrm{do}(a_t = \mathfrak{a}))\mathrm{d}u \tag{7}$$

$$= \int_u p(r_t|z_t, a_t = \mathfrak{a}, u)p(u)\mathrm{d}u, \tag{8}$$

---

2. To give some intuition, take for example the case of kidney stones. One prefers treatment b when considering the overall probability of recovery. By contrast, one chooses treatment a when considering the recovery rate when conditioning on each possible kidney stone size. The optimal treatment is treatment a in this example. The *do*-operator in Equation (2) allows us to obtain the correct solution in this type of problems with confounders. By contrast, traditional RL methods will fail in such settings since they consider only conditional probabilities (e.g., the overall probability of recovery) instead of interventional probabilities, as the *do*-operator does.

where Equation (8) is obtained by applying the rules of *do*-calculus to the causal graph in Figure 2 (Pearl, 2009). We also can use the back-door criterion to directly obtain Equation (8). Equation (8) shows that $p(r_t|z_t, \mathrm{do}(a_t = \mathfrak{a}))$ can be identified from the joint distribution $p(u, \vec{z}, \vec{x}, \vec{a}, \vec{r})$, because, through Equation (8), the interventional probability $p(r_t|z_t, \mathrm{do}(a_t = \mathfrak{a}))$ is converted to observation probabilities with regard to $p(u, \vec{z}, \vec{x}, \vec{a}, \vec{r})$. In other words, if we can recover $p(u, \vec{z}, \vec{x}, \vec{a}, \vec{r})$ then we also can recover $p(r_t|z_t, \mathrm{do}(a_t = \mathfrak{a}))$.

Now the problem is reduced to whether or not we can estimate $p(u, \vec{z}, \vec{x}, \vec{a}, \vec{r})$ from observations of $(\vec{x}, \vec{a}, \vec{r})$. Fortunately, it is possible, because a number of works have shown that one can use the knowledge inferred from the joint distribution between proxy variables and confounders to adjust for the hidden confounders (Louizos et al., 2017; Edwards et al., 2015; Kuroki and Pearl, 2014; Miao et al., 2018; Pearl, 2012; Wooldridge, 2009). Here, we focus only on one possible case presented in Figure 1 of (Louizos et al., 2017) (see Appendix D), because their result can be directly used to show that the joint distribution $p(u, \vec{z}, \vec{x}, \vec{a}, \vec{r})$ can be approximately recovered solely from the observations $(\vec{x}, \vec{a}, \vec{r})$. The main idea is that our model can be factorized into two types of 4-tuple components, each of which can be proved using the same method in (Louizos et al., 2017). More precisely, in our model, at each time step the 4-tuple (z_t, x_t, a_t, r_{t+1}) is exactly the same as the 4-tuple (Z, X, t, y) shown in Figure 1 of (Louizos et al., 2017). Since it has been proved in (Louizos et al., 2017) that the joint distribution $p(Z, X, t, y)$ can be recovered from observations of (X, t, y), $p(z_t, x_t, a_t, r_{t+1})$ can be also approximately recovered only from the observations (x_t, a_t, r_{t+1}). Likewise, the joint distribution over the other 4-tuple (u, x_t, a_t, r_{t+1}) can be recovered from (x_t, a_t, r_{t+1}) in the same manner (see Appendix E). Applying this rule repeatedly to the sequential data, we are finally able to approximately recover $p(u, \vec{z}, \vec{x}, \vec{a}, \vec{r})$ solely from the observations $(\vec{x}, \vec{a}, \vec{r})$. Hence, in the present paper we may reasonably assume that the joint distribution $p(u, \vec{z}, \vec{x}, \vec{a}, \vec{r})$ can be approximately recovered solely from the observations $(\vec{x}, \vec{a}, \vec{r})$.

We estimate the joint distribution $p(u, \vec{z}, \vec{x}, \vec{a}, \vec{r})$ using Variational Auto-Encoders (VAEs) (Kingma and Welling, 2013), which can represent a very large class of distributions with latent variables and are easily implemented by solving an optimization problem (Tran et al., 2015; Louizos et al., 2017). However, VAEs do not guarantee that the true model can be identified through learning (Louizos et al., 2017). For example, it might occur that different model parameters could equally fit the same observed data as long as the values of the latent confounders are adjusted accordingly (D'Amour, 2018). Despite this, identification is possible in both our model and the one described by Louizos et al. (2017) under specific conditions (Louizos et al., 2017; Kuroki and Pearl, 2014; Miao et al., 2018; Pearl, 2012; Allman et al., 2009). In addition to this, in our model, each of the z_t and u have at least T proxy variables $\{x_t\}_{t=1,\ldots,T}$, where z_t can be approximately viewed as hidden states in a hidden Markov model. In this case, Allman et al. (2009) show that such a model can be identified if the latent confounders z_t and u are assumed to be categorical. However, in practice there is no guarantee for this to be the case. Hence, we prefer to use VAEs, which make substantially weaker assumptions about the data generating process and the structure of the hidden confounders (Louizos et al., 2017). Despite the lack of general identifiability results for VAEs, we empirically show that our approach is useful for learning a better policy in the presence of confounding.

In the model from Figure 2 there are two types of confounders: the time-independent confounder u and the time-dependent confounders $\{z_t\}_{t=1,\dots,T}$, playing different roles in the model. We refer to the former as "nuisance confounders" and the latter as "advantage confounders". The nuisance confounder u will negatively affect the whole course of treatment and, therefore, should be adjusted for. In the example of kidney stones, the existence of the confounder (i.e., the size of stones) will lead to a wrong treatment if not adjusted for. By contrast, the advantage confounders $\{z_t\}$ act as state variables and, in principle, they are not supposed to be adjusted for because in RL we aim to take optimal actions when conditioning on the current state[3].

3.4 Learning

The nonlinear functions parametrized by neural networks make inference intractable. Because of this, we learn the parameters of the model θ_k by using variational inference together with an inference model, a neural network which approximates the intractable posterior distributions (Rezende et al., 2014; Kingma and Welling, 2013; Krishnan et al., 2015). We now review how to learn a simple latent variable model $p_\theta(x|z)$ using variational inference. In this simple model x stands for the observational variables and z for the latent variables. By using the variational principle, we introduce an approximate posterior distribution $q_\phi(z|x)$ to obtain the following lower bound on the model's marginal likelihood:

$$\log p_\theta(x) \geq \mathop{\mathbb{E}}_{q_\phi(z|x)} \left[\log p_\theta(x|z)\right] - \text{KL}\left(q_\phi(z|x)||p_\theta(z)\right), \tag{9}$$

where we have used Jensen's inequality, and $\text{KL}(\cdot||\cdot)$ denotes the Kullback-Leibler divergence between two distributions and ϕ are the parameters of the inference model $q(z|x)$.

3.4.1 Variational Lower Bound

By directly applying the lower bound in Equation (9) to our model, we obtain

$$\log p_\theta(\vec{x}, \vec{a}, \vec{r}) \geq$$
$$\mathop{\mathbb{E}}_{q_\phi(\vec{z}, u|\vec{x}, \vec{a}, \vec{r})} \left[\log p_\theta(\vec{x}, \vec{a}, \vec{r}, \vec{z}, u)\right] - \text{KL}\left(q_\phi(\vec{z}, u|\vec{x}, \vec{a}, \vec{r})||p_\theta(\vec{z}, u)\right) \equiv \mathcal{L}(\vec{x}, \vec{a}, \vec{r}; \theta, \phi). \tag{10}$$

By using the Markov property of our model, we can factorize the full joint distribution in the following way:

$$p_\theta(\vec{x}, \vec{a}, \vec{r}, \vec{z}, u) = p(u)p(z_1)\left[\prod_{t=1}^{T} p(x_t|z_t, u)p(a_t|z_t, u)p(r_{t+1}|z_t, a_t, u)\right]$$
$$\left[\prod_{t=2}^{T} p(z_t|z_{t-1}, a_{t-1})\right]. \tag{11}$$

In addition, for simplicity, we also have a similar factorization assumption in the posterior approximation for $\vec{z}$ and u:

$$q_\phi(\vec{z}, u|\vec{x}, \vec{a}, \vec{r}) = q(u|\vec{x}, \vec{a}, \vec{r})q(z_1|\vec{x}, \vec{a}, \vec{r})\prod_{t=2}^{T} q(z_t|z_{t-1}, \vec{x}, \vec{a}, \vec{r}). \tag{12}$$

---

3. Another reason why we do not need to adjust for z_t has been discussed in Section 3.2.

Combining equations (10), (11) and (12) yields

$$\mathcal{L}(\vec{x}, \vec{a}, \vec{r}; \theta, \phi) = \sum_{t=1}^{T} \mathop{\mathbb{E}}_{\substack{z_t \sim q(z_t | z_{t-1}, \vec{x}, \vec{a}, \vec{r}) \\ u \sim q(u | \vec{x}, \vec{a}, \vec{r})}} \left[\log p(x_t | z_t, u) + \log p(a_t | z_t, u) + \log p(r_{t+1} | z_t, a_t, u) \right]$$
$$- \mathrm{KL}\left(q(u | \vec{x}, \vec{a}, \vec{r}) || p(u) \right) - \mathrm{KL}\left(q(z_1 | \vec{x}, \vec{a}, \vec{r}) || p(z_1) \right)$$
$$- \sum_{t=2}^{T} \mathop{\mathbb{E}}_{z_{t-1} \sim q(z_{t-1} | z_{t-2}, \vec{x}, \vec{a}, \vec{r})} \left[\mathrm{KL}\left(q(z_t | z_{t-1}, \vec{x}, \vec{a}, \vec{r}) || p(z_t | z_{t-1}, a_{t-1}) \right) \right], \quad (13)$$

where we omit the subscripts θ and ϕ. A more detailed derivation can be found in Appendix A. Equation (13) is differentiable with respect to the model parameters θ and the inference parameters ϕ and, by using the reparametrization trick (Kingma and Welling, 2013), we can directly apply backpropagation to optimize this objective.

3.4.2 INFERENCE MODEL

From the factorization in Equation (12), we can see that there are two types of inference models: $q(u | \vec{x}, \vec{a}, \vec{r})$ and $q(z_t | z_{t-1}, \vec{x}, \vec{a}, \vec{r})$. Similar to the generative model in Section 3.2, we also parametrize both of them as Gaussian:

$$q(u | \vec{x}, \vec{a}, \vec{r}) = \mathcal{N}\left(u | \hat{\mu}_t^u, \hat{\sigma}_t^{u\,2} \right); \quad \hat{\mu}_t^u = f_9(\vec{x}, \vec{a}, \vec{r}), \quad \hat{\sigma}_t^{u\,2} = f_{10}(\vec{x}, \vec{a}, \vec{r}); \quad (14)$$

$$q(z_t | z_{t-1}, \vec{x}, \vec{a}, \vec{r}) = \mathcal{N}\left(z_t | \hat{\mu}_t^z, \hat{\sigma}_t^{z\,2} \right); \quad \hat{\mu}_t^z = f_{11}(z_{t-1}, \vec{x}, \vec{a}, \vec{r}), \quad \hat{\sigma}_t^{z\,2} = f_{12}(z_{t-1}, \vec{x}, \vec{a}, \vec{r}). \quad (15)$$

In fact, as shown in Equation (12), $q(\vec{z} | \vec{x}, \vec{a}, \vec{r})$ can be factorized as the product of $q(z_t | z_{t-1}, \vec{x}, \vec{a}, \vec{r})$ for $t = 1, \ldots, T$. By using the Markov property of our model, we have

$$z_t \perp\!\!\!\perp x_1, \ldots, x_{t-1}, a_1, \ldots, a_{t-2}, r_2, \ldots, r_t | z_{t-1}. \quad (16)$$

Therefore, we can further simplify each of these terms as follows:

$$q(z_t | z_{t-1}, \vec{x}, \vec{a}, \vec{r}) = q(z_t | \underbrace{z_{t-1}, a_{t-1}}_{\text{past}}, \underbrace{x_t, a_t, r_{t+1}}_{\text{present}}, \underbrace{x_{t+1}, a_{t+1}, r_{t+2}, \ldots, x_T, a_T, r_{T+1}}_{\text{future}}). \quad (17)$$

Equation (17) tells us that z_t depends on z_{t-1} and all the current and future observed data in $(\vec{x}, \vec{a}, \vec{r})$. Meanwhile, the conditional independence above means that z_{t-1} contains all the historical data. Therefore, it is natural to calculate z_t based on the whole sequence of data. This can be done using recurrent neural networks (RNNs). Inspired by Krishnan et al. (2015, 2017), we similarly choose a bi-directional LSTM (Zaremba and Sutskever, 2014) to parameterize f_{11} and f_{12} in Equation (15). Since Equation (14) has the same structure as Equation (15), f_9 and f_{10} are parameterized by a bi-directional LSTM as well. More details about the architecture can be found in Appendix C.

Note that to generate data from the model, at any time step t, we require to know a_t and r_{t+1} before inferring the distribution over z_t when conditioning on the observed x_t. Hence, we need to introduce two auxiliary distributions to perform counterfactual reasoning. This

corresponds to the problem of predicting x_{t+1} given an x_t unseen in the training set. To be more precise, we have

$$q(a_t|x_t) = \mathcal{N}\left(a_t|\hat{\mu}_t^{\mathfrak{a}}, \hat{\sigma}_t^{\mathfrak{a}^2}\right); \quad \hat{\mu}_t^{\mathfrak{a}} = f_{13}(x_t), \quad \hat{\sigma}_t^{\mathfrak{a}^2} = f_{14}(x_t); \tag{18}$$

$$q(r_{t+1}|x_t, a_t) = \mathcal{N}\left(r_{t+1}|\hat{\mu}_t^{\mathfrak{r}}, \hat{\sigma}_t^{\mathfrak{r}^2}\right); \quad \hat{\mu}_t^{\mathfrak{r}} = f_{15}(x_t, a_t), \quad \hat{\sigma}_t^{\mathfrak{r}^2} = f_{16}(x_t, a_t), \tag{19}$$

where f_{13}, f_{14}, f_{15}, and f_{16} are also parameterized by neural networks. To estimate the parameters of these inference models, we will add extra terms in the variational lower bound given by Equation (13):

$$\mathcal{L}_{\mathrm{DRL}}(\vec{x}, \vec{a}, \vec{r}; \theta, \phi) = \mathcal{L}(\vec{x}, \vec{a}, \vec{r}; \theta, \phi) + \sum_{t=1}^{T} \left(\log q(a_t|x_t) + \log q(r_{t+1}|x_t, a_t)\right). \tag{20}$$

3.5 Deconfounding RL Algorithms

We have now all the building blocks for our DRL algorithm. Once our model is learned from the observational data, it can be directly used as a dynamic environment like those in OpenAI Gym (Brockman et al., 2016). We can exploit the learned model to generate rollouts for policy evaluation and optimization. In practice, Equation (8) is approximated using the Monte Carlo method as follows:

$$p(r_t|z_t = \mathfrak{z}, \mathrm{do}(a_t = \mathfrak{a})) = \frac{1}{N} \sum_{i=1}^{N} p(r_t|z_t = \mathfrak{z}, a_t = \mathfrak{a}, u_i) \quad u_i \sim p(u), \tag{21}$$

where N is the sample size from the prior $p(u)$. Given observational data, we sample u instead from the approximate posterior $q(u|\vec{x}, \vec{a}, \vec{r})$ which we compute using the inference network presented in Section 3.4.2.

By using this deconfounding reward function, it is straightforward to extend traditional RL algorithms to their corresponding deconfounding version. In this paper, we select and extend one representative RL algorithm: the Actor-Critic method (Sutton et al., 1998). Nevertheless, our methodology can be used to extend other algorithms as well in a straightforward manner.

Deconfounding Actor-Critic Methods   The actor-critic method is a policy-based RL method directly parameterizing a policy function $\pi(a|z; \theta)$. The goal is to reduce variance in the estimate of the policy gradient by subtracting a learned function of the state, known as a baseline, from the return. The learned value function $V(z; \phi)$ is commonly used as the baseline. Taking into consideration that the return is estimated by $Q(z, a; \phi_Q)$, we can write the gradient of the actor-critic loss function at step time t as

$$\nabla J(\theta) = \mathbb{E}_\pi\left[(Q(z_t, a_t; \phi_Q) - V(z_t; \phi_V))\nabla_\theta \ln \pi(a_t|z_t; \theta)\right], \tag{22}$$

where $Q(z_t, a_t; \phi_Q) - V(z_t; \phi_V)$ is an estimate of the advantage of action a_t in state z_t. In practice, $Q(z_t, a_t; \phi_Q)$ is usually replaced with the one-step return, that is, $r_{t+1} + V(z_{t+1}; \phi_V)$. The crucial difference in deconfounding actor-critic methods is to use $r_{t+1} \sim p(r_{t+1}|z_t, \mathrm{do}(a_t))$ given by Equation (21), as opposed to the $r_{t+1} \sim p(r_{t+1}|z_t, a_t)$ used by vanilla actor-critic methods.

3.6 Training

As mentioned previously, DRL consists of two steps: learning a deconfounding model and optimizing a policy based on the learned deconfounding model. In step 1 we learn the model by optimizing the objective given by Equation (20). Once the deconfounding model is learned, we have an estimate of the state transition function $p(z_t|z_{t-1}, a_{t-1})$, as given by the model, and can also calculate the deconfounding reward function $p(r_t|z_t, do(a_t))$ according to Equation (21). In step 2, we treat the learned deconfounding model as an RL environment like CartPole in OpenAI Gym and generate trajectories/rollouts using the estimated state transition function and deconfounding reward function. These trajectories/rollouts are then used to train the policy by following the gradient given by Equation (22).

3.7 Implementation Details

We used Tensorflow (Abadi et al., 2016) for the implementation of our model and the proposed DRL algorithm. Optimization was done using Adam (Kingma and Ba, 2014). Unless stated otherwise, the setting of all hyperparameters and network architectures can be found in Appendix G.

To assess the equality of the learned model, we performed two types of tasks: reconstruction and counterfactual reasoning. The reconstructions were performed by first feeding an input sequence $\vec{x}$ into the learned inference network, then sampling from the resulting posterior distribution over $\vec{z}$ according to Equation (15), and finally feeding those samples into the generative network described in Equation (3) to reconstruct the original observed sequence $\vec{x}$. The counterfactual reasoning, that is, the prediction of x_{t+1} given an x_t that we have not seen during training, was executed in four steps:

1) Given the new x_t, we estimate a_t and r_{t+1} using equations (18) and (19).

2) Once we have x_t, a_t, and r_{t+1}, we estimate z_t using Equation (15).

3) Given the estimated z_t and another uniformly randomly selected a'_t, we can directly compute z_{t+1} from Equation (6).

4) The final step is to reconstruct x_{t+1} from z_{t+1} and u according to Equation (3).

By repeating the last two steps, we can counterfactually reason out a sequence of data.

We may also be interested in estimating the approximate posterior over the confounder u from observed data. We consider two possible scenarios for this. In the easy one we are given the observed data $(\vec{x}, \vec{a}, \vec{r})$ and we estimate the posterior by using Equation (14). The more challenging scenario involves estimating the posterior from only $\vec{x}$ and no action or reward data. In this case, we follow the same steps used in the task of counterfactual reasoning. We first compute a_t and r_{t+1} for $t = 1$ to T using Equations (18) and (19), and then estimate the posterior through Equation (14).

4. Experimental Results

The evaluation of any method that deals with confounders is always challenging due to a lack of real-world benchmark problems with known ground truth. Furthermore, little work

has been done before on the task of deconfounding RL. All this creates difficulties in the evaluation of our algorithms and motivates us to develop several new benchmark problems by revising the MNIST dataset (LeCun et al., 1998) and by revising two environments in OpenAI Gym (Brockman et al., 2016), CartPole and Pendulum.

4.1 New Confounding Benchmarks

We now describe three new confounding benchmark problems, all of them including a single binary confounder. The procedure to create these benchmarks is inspired by Krishnan et al. (2015), who synthesized a dataset mimicking healthcare data under realistic conditions (e.g., noisy laboratory measurements, surgeries and drugs affected by patient age and gender, etc.). The data used is from either popular RL environments in OpenAI Gym such as Pendulum and CartPole or popular machine learning datasets such as MNIST.

Confounding Pendulum and CartPole   We revised two environments in OpenAI Gym (Brockman et al., 2016): Pendulum and CartPole. To obtain a confounding version of Pendulum, we selected 100 different screen images of Pendulum and created a synthetic problem in which actions are joint effort[4] with values between -2 and 2. We added 20% bit-flip noise to each image and then, a random policy confounded with a binary factor was used to select the action applied at each time step. This is repeated multiple times to produce a large number of 5-step sequences of noisy images. In each generated sequence, one block with three consecutive squares (2×2 pixels) is superimposed with the top-left corner of the images in a random starting location. These squares are intended to be analogous to seasonal flu or other ailments that a patient could exhibit, which are independent of the actions taken and which last several time steps (Krishnan et al., 2015). The goal is to show that our model can learn long-range patterns since these play an important role in medical applications. We treat the generated sequences of images as the observations $\vec{x}$. The training, validation and test sets respectively comprise $140,000$, $28,000$ and $28,000$ sequences of length five.

The key characteristic of confounding Pendulum is the relationship between confounder, actions, and rewards. For simplicity of notation, we denote the confounder by u, the action by a, and the reward by r. In this case, u is a binary variable mimicking socio-economic status (i.e., the rich and the poor). The range of actions $a \in [-2, 2]$ is grouped into two categories T_1 (i.e., $1 \le |a| \le 2$) and T_2 (i.e., $0 \le |a| \le 1$) representing different treatments[5]. The treatment selection depends on u, that is, u determines the probabilities of choosing T_1 and T_2. The reward is defined as

$$r = r_{\mathrm{o}} + r_{\mathrm{c}}, \tag{23}$$

where r_o is the original reward[6] in Pendulum as a function of a, and r_c is the extra reward as a function of both u and a, defined by a two-component Gaussian mixture:

$$r_c \sim \sum\nolimits_{r_c \in \{R_1, R_2\}} \pi(r_c|a, u) \mathcal{N}(\mu_{r_c}, \sigma^2), \tag{24}$$

---

4. More details can be found at `https://github.com/openai/gym/wiki/Pendulum-v0`.

5. Note that T_1 is a better treatment than T_2 in our setting described in Appendix F.

6. In fact, here r_o is a function of both a and a state, and we mention only a to emphasize that the confounder affects the action.

with σ and μ_{r_c} fixed and mixing coefficient $\pi(r_c|a, u)$ determined by both u and a. More details are available in Appendix F.

Obviously, in the definition above, r depends on a and u. Furthermore, u has an influence on x_t through a_{t-1} and z_t, meaning that x_t contains information about u and, therefore, can be viewed as a proxy variable for the confounder. We assume that the influence between the confounder, action and reward is stochastic while generating the data. The reason for this is that, in practice, we do not have an oracle that tells us which treatment is better so we might make wrong decisions sometimes. For example, take the case of kidney stones. Even though treatment a is better than treatment b, there are still some patients choosing treatment b with positive probability in each category. All the details about the data generation process can be found in Appendix F, where a straightforward analogy is provided as well.

The confounding CartPole problem is implemented in the exact same manner, except for the action which is now binary and can be naturally divided into two categories[7]. Accordingly, the binary confounder determines the probabilities of choosing which of two actions, and the probabilities are set to the same value as those in the confounding Pendulum given in Appendix F.

Confounding MNIST We follow the same procedure to obtain a confounding MNIST problem. However, the definitions of action and the original reward term are now different. Similar to the Healing MNIST dataset (Krishnan et al., 2015), the actions encode rotations of digit images with each entry in $\vec{a}$ satisfying $(-\frac{\pi}{4} \le a \le +\frac{\pi}{4})$ and the actions being divided into two categories $(\frac{\pi}{8} \le |a| \le \frac{\pi}{4})$ and $(0° \le |a| < \frac{\pi}{8})$ according to the confounder u. The original reward term r_o is defined as the minus degree between the upright position and the position that the digit rotates to. For example, if the digit rotates to the position of 3 o'clock or 9 o'clock, then both rewards are $-\frac{\pi}{2}$.

4.2 Performance Analysis of the Deconfounding Model

We assess the performance of the deconfounding model from Figure 2, denoted by M_{decon}, and compare it with a similar alternative model that does not include the confounder u and that is denoted by M_{alt}. We train M_{decon} by optimizing Equation (20) but train M_{alt} using a different loss function which excludes the confounder u and whose full derivation can be found in Appendix B. Both models are separately trained in a minibatch manner on the training set of the confounding dataset (i.e., $140K$ image sequences of length five). Afterwards, following the steps depicted in Section 3.7, we use each trained model to perform the reconstruction task on the training set, and both reconstruction and counterfactual reasoning tasks on the testing set (i.e., $28K$ image sequences of length five).

Figure 3 presents a comparison of M_{decon} and M_{alt} in terms of reconstruction and counterfactual reasoning on confounding Pendulum. The second row is based on M_{decon} whilst the top row comes from M_{alt}. The results generated by the deconfounding model are superior to those produced by the model not taking into account the confounder. To be more specific, as shown in the zoom of samples on the bottom row, M_{alt} generates more blurry images than M_{decon} because, without modelling the confounder u, M_{alt} is forced to average over its multiple latent states, resulting in more blurry samples.

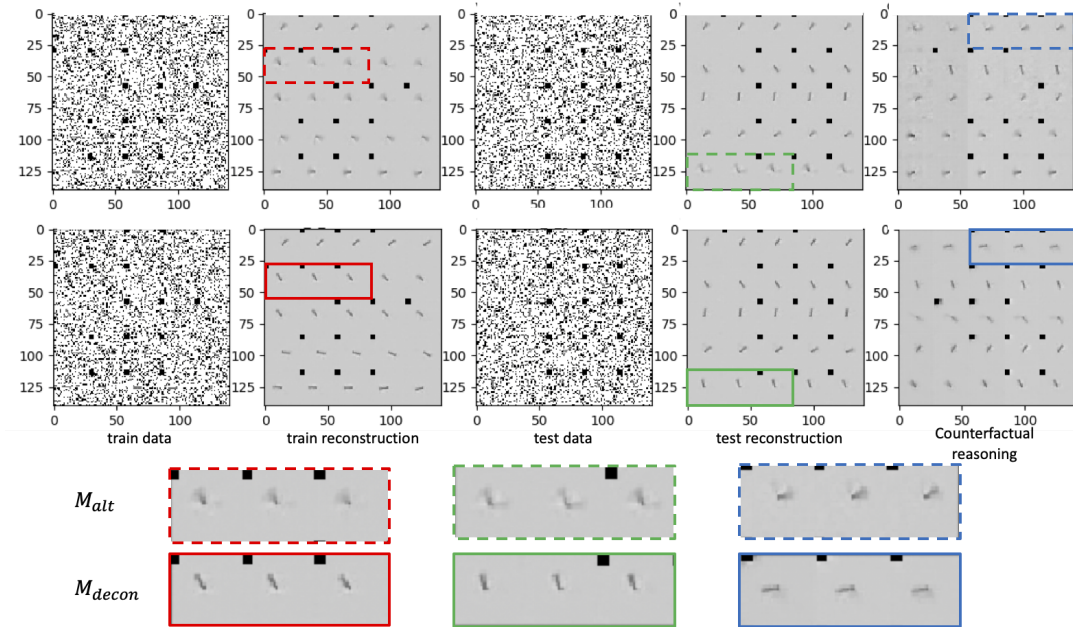7. More details can be found at https://github.com/openai/gym/wiki/CartPole-v0.

Figure 3: Reconstruction and counterfactual reasoning on the confounding Pendulum dataset. Top row: results from the model without the confounder u (M_{alt}). Second row: results from the model with the confounder u (M_{decon}). The last two rows are the zoom of samples selected in the same positions from the top row (dashed boxes for M_{alt}) and the second row (solid boxes for M_{decon}), respectively. These results show that the model taking into account confounding performs better in every task producing less blurry images.

We obtain similar results regarding the samples produced by M_{alt} and M_{decon} on the confounding MNIST dataset, as shown in Figure 4. Looking closely at the squares on the generated digit samples (inside a yellow box), we observe that M_{alt} generates non-consecutive white squares in the task of counterfactual reasoning, which does not really make sense because only consecutive patterns appear in the training set. The generated images are also more blurry. By contrast, this does not occur on the samples from M_{decon}, showing that our deconfounding model is able to cope with long-range patterns and describe the data better.

Figure 5 visualizes approximate posterior samples of the 2-dimensional confounder u. We can see that, although the prior distribution of the confounder is assumed to be a factorized standard Gaussian distribution, the model still identifies two clear clusters from the data because u is originally a binary variable. This demonstrates that our model can learn confounders even if the assumed prior is not that accurate.
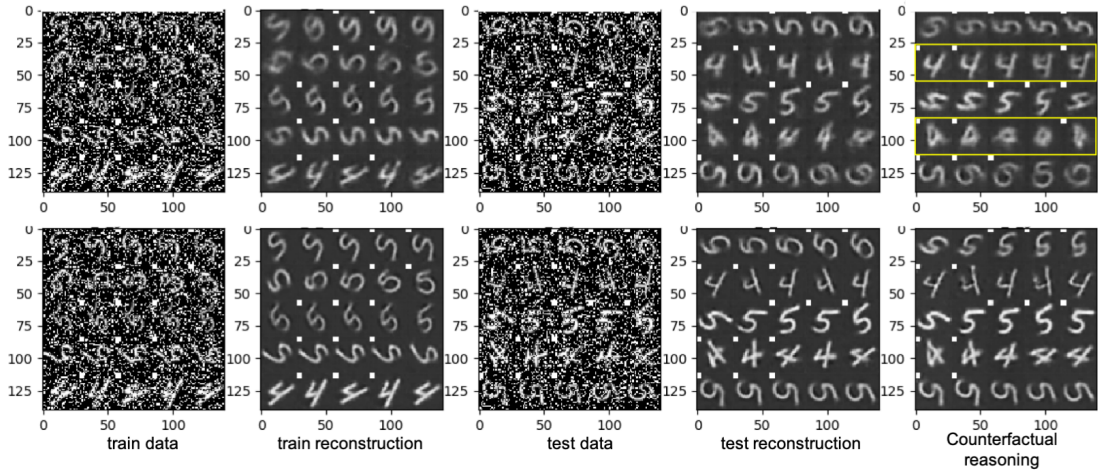
Figure 4: Reconstruction and counterfactual reasoning on the confounding MNIST dataset. Top row: results from the model without the confounder u (M_{alt}). Bottom row: results from the model with the confounder u (M_{decon}). Note that in the task of counterfactual reasoning, the second and forth sequences of the sample produced by M_{alt}, boxed in yellow, have non-consecutive white squares and contain more blurry images. This does not really make sense because only consecutive patterns appear in the training set. By contrast, this kind of non-consecutive white squares does not occur on the samples from M_{decon}.
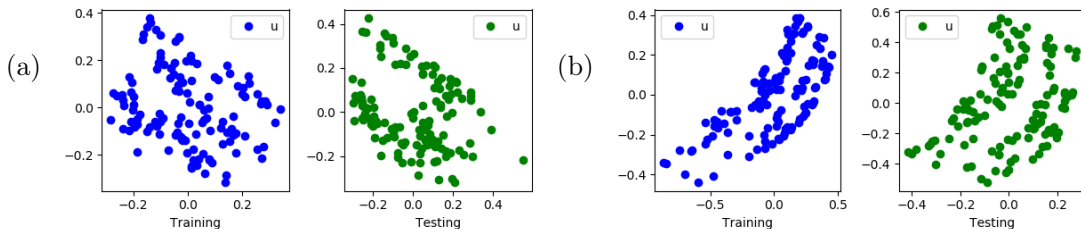
Figure 5: (a) Plot of 128 data points sampled from the posterior approximate distribution of u on the confounding MNIST dataset. (b) Plot of 128 data points sampled from the posterior approximate distribution of u on the confounding CartPole dataset. We can see that, given a binary confounder u, our model still identifies two clear clusters from the data even if the assumed prior over u is a factorized standard Gaussian distribution.

4.3 Comparison of RL Algorithms

In this section, we evaluate the proposed deconfounding actor-critic (AC) method by comparing with its vanilla version on confounding Pendulum. In the vanilla AC method, given a learned M_{alt}, we optimize the policy by calculating the gradient presented in Equation (22)
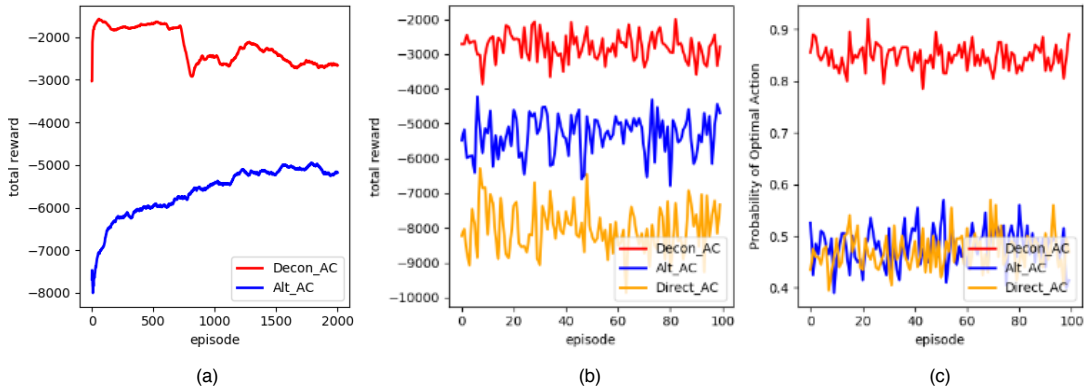
Figure 6: Comparison of vanilla (Alt_AC), direct (Direct_AC), and deconfounding (Decon_AC) Actor-Critic methods on the confounding Pendulum problem. (a) total reward over 1500 episodes in the training phase. (b) total reward over 100 episodes in the testing phase. (c) Probability of optimal action over 100 episodes, corresponding to (b).

on the basis of the trajectories/rollouts generated through M_{alt}. Equation (22) involves two functions: $V(z_t; \phi_V)$ and $\pi(a_t|z_t; \theta)$, both of which are represented using neural networks and their corresponding hyperparameters can be found in Appendix G. It is worth noting that, in this vanilla case, each reward r_{t+1} is sampled from the conditional distribution $p(r_{t+1}|z_t, a_t)$. By contrast, the proposed deconfounding AC method uses M_{decon} instead. The optimizaiton of the policy is again performed using the gradient from Equation (22). However, for the deconfounding AC approach, we use trajectories/rollouts generated by M_{decon} in which each reward r_{t+1} is obtained using the interventional distribution $p(r_{t+1}|z_t, \mathrm{do}(a_t))$ computed according to Equation (21). For completeness, we also compare with the direct AC method in which the AC method is directly trained on the training data instead of the trajectories/rollouts.

In the training phase, we respectively run the vanilla AC and the deconfounding AC algorithms over 2000 episodes with 200 time steps each. For the fair comparison, we also run the direct AC algorithm over 2000 episodes, each of which has 200 pairs of state-action-reward randomly selected from the training data. In order to reduce non-stationarity and to decorrelate updates, the generated data is stored in an experience replay memory and then randomly sampled in a batch manner (Mnih et al., 2013; Riedmiller, 2005; Schulman et al., 2015; Van Hasselt et al., 2016). We summarize all the rewards produced during the rollouts in each episode and further average the summarized rewards over a window of 100 episodes to obtain a smoother curve. Figure 6(a) shows that our deconfounding AC algorithm performs significantly better than the vanilla AC algorithm in the confounded environment. Here, the direct AC algorithm is not included because it does not generate rollouts.

In the testing phase, we first randomly select 100 samples from the testing set, each starting a new episode, and then use the learned policies to generate trajectories over 200 time steps as we did during training. From the resulting 100 episodes, we plot the total

reward obtained by each method, see in Figure 6(b), and also plot the percentage of times that the optimal action T_1 is chosen in each episode, as shown in Figure 6(c). Figure 6(b) shows that our deconfounding AC obtains on average much higher reward at test time than the baselines. Figure 6(c) further tells us that our deconfounding AC is much more likely to choose the optimal action at each time step, whilst the vanilla AC and the direct AC make a wrong decision more than half of the times.

5. Related Work

Krishnan et al. (2015, 2017) used deep neural networks to construct nonlinear state space models and leveraged a structured variational approximation parameterized by recurrent neural networks to mimic the posterior distribution. Levine (2018) reformulated RL and control problems using probabilistic inference, which allows us to bring to bear a large pool of approximate inference methods, and flexibly extend the models. Raghu et al. (2017a,b) exploited continuous state-space models and deep RL to obtain improved treatment policies for septic patients from observational data. Gottesman et al. (2018) discussed problems when evaluating RL algorithms in observational health setting. However, all the works mentioned above do not take into account confounders in the proposed models.

Louizos et al. (2017) attempted to learn individual-level causal effects from observational data in the non-temporal setting. They also used a variational auto-encoder to estimate the unknown confounder given a causal graph. Paxton et al. (2013) developed predictive models based on electronic medical records without using causal inference. Saria et al. (2010) proposed a nonparametric Bayesian method to analyze clinical temporal data. Soleimani et al. (2017) represented the treatment response curves using linear time-invariant dynamical systems. This provides a flexible approach to modeling response over time. Although the latter two works model sequential data, they both do not consider RL or causal inference.

Bareinboim et al. (2015) considered the problem of bandits with unobserved confounders. Sen et al. (2016) and Ramoly et al. (2017) further studied contextual bandits with latent confounders. Forney et al. (2017) circumvented some problems caused by unobserved confounders in multi-armed bandits by using counterfactual-based decision making. Zhang and Bareinboim (2017) leveraged causal inference to tackle the problem of transferring knowledge across bandit agents. Unlike our method, all these works are restricted to bandits, which corresponds to a simplified RL setting without state transitions.

Last but not least, we want to mention some connections between our method and partially observable Markov decision processes (POMDPs). To the best of our knowledge, existing work on POMDPs has not considered problems with confounders yet. Apart from the confounding part, in POMDPs the observation only provides partial information about the actual state, so the agent in POMDPs does not necessarily know which actual state it is in. By contrast, for convenience in practice, we simplify this setting and assume that the observation provides all the information about the actual state the agent is in but with some noise. Hence we only need to denoise the observation to obtain the actual state. In this sense, a more related work to our model is probably the world model (Ha and Schmidhuber, 2018), because both models used variational inference to estimate the actual state from the observation. If we treat our model as POMDPs with the confounding bias, it will make our model more complicated but it is worth exploring in the future.

As far as we are concerned, this is the first attempt to build a bridge between confounding and the full RL problem with observational data.

6. Conclusion and Future Work

We have introduced deconfounding reinforcement learning (DRL), a general method for addressing reinforcement learning problems with observational data in which hidden confounders affect observed rewards and actions. We have used DRL to obtain deconfounding variants of actor-critic methods and showed how these new variants perform better than the original vanilla algorithms on new confounding benchmark problems. In the future, we will collaborate with hospitals and apply our approach to real-world medical datasets. We also hope that our work will stimulate further investigation of connections between causal inference and RL.

References

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: a system for large-scale machine learning. In *OSDI*, volume 16, pages 265–283, 2016.

Ahmed M Alaa and Mihaela van der Schaar. Bayesian nonparametric causal inference: Information rates and learning algorithms. *IEEE Journal of Selected Topics in Signal Processing*, 12(5):1031–1046, 2018.

Ahmed M Alaa, Michael Weisz, and Mihaela van der Schaar. Deep counterfactual networks with propensity-dropout. *arXiv preprint arXiv:1706.05966*, 2017.

Elizabeth S Allman, Catherine Matias, John A Rhodes, et al. Identifiability of parameters in latent structure models with many observed variables. *The Annals of Statistics*, 37(6A): 3099–3132, 2009.

Joshua D Angrist and Jörn-Steffen Pischke. *Mostly harmless econometrics: An empiricist's companion*. Princeton university press, 2008.

Onur Atan, William R Zame, Qiaojun Feng, and Mihaela van der Schaar. Constructing effective personalized policies using counterfactual inference from biased data sets with many features. *arXiv preprint arXiv:1612.08082*, 2016.

Elias Bareinboim, Andrew Forney, and Judea Pearl. Bandits with unobserved confounders: A causal approach. In *Advances in Neural Information Processing Systems*, pages 1342–1350, 2015.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016.

Lars Buesing, Theophane Weber, Yori Zwols, Nicolas Heess, Sebastien Racaniere, Arthur Guez, and Jean-Baptiste Lespiau. Woulda, coulda, shoulda: Counterfactually-guided policy search. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=BJG0voC9YQ.

Alexander D'Amour. (non-)identification in latent confounder models. https://www.alexdamour.com/blog/public/2018/05/18/non-identification-in-latent-confounder-models/, 2018.

Jessie K Edwards, Stephen R Cole, and Daniel Westreich. All your data are always missing: incorporating bias due to measurement error into the potential outcomes framework. *International journal of epidemiology*, 44(4):1452–1459, 2015.

Ronald Aylmer Fisher. The design of experiments. 1935.

Andrew Forney, Judea Pearl, and Elias Bareinboim. Counterfactual data-fusion for online reinforcement learners. In *International Conference on Machine Learning*, pages 1156–1164, 2017.

Omer Gottesman, Fredrik Johansson, Joshua Meier, Jack Dent, Donghun Lee, Srivatsan Srinivasan, Linying Zhang, Yi Ding, David Wihl, Xuefeng Peng, et al. Evaluating reinforcement learning algorithms in observational health settings. *arXiv preprint arXiv:1805.12298*, 2018.

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

MA Hernán and JM Robins. *Causal Inference*. Boca Raton: Chapman and Hall/CRC, forthcoming, 2018.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Rahul G Krishnan, Uri Shalit, and David Sontag. Deep kalman filters. *arXiv preprint arXiv:1511.05121*, 2015.

Rahul G Krishnan, Uri Shalit, and David Sontag. Structured inference networks for nonlinear state space models. In *AAAI*, pages 2101–2109, 2017.

Manabu Kuroki and Judea Pearl. Measurement bias and effect restoration in causal inference. *Biometrika*, 101(2):423–437, 2014.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Sergey Levine. Reinforcement learning and control as probabilistic inference: Tutorial and review. *arXiv preprint arXiv:1805.00909*, 2018.

Qing Liu, Katharine Henry, Yanbo Xu, and Suchi Saria. Using causal inference to estimate what-if outcomes for targeting treatments.

Christos Louizos, Uri Shalit, Joris M Mooij, David Sontag, Richard Zemel, and Max Welling. Causal effect inference with deep latent-variable models. In *Advances in Neural Information Processing Systems*, pages 6446–6456, 2017.

Gangadharrao Soundaryarao Maddala and Kajal Lahiri. *Introduction to econometrics*, volume 2. Macmillan New York, 1992.

PE Marik. The demise of early goal-directed therapy for severe sepsis and septic shock. *Acta Anaesthesiologica Scandinavica*, 59(5):561–567, 2015.

Wang Miao, Zhi Geng, and Eric J Tchetgen Tchetgen. Identifying causal effects with proxy variables of an unmeasured confounder. *Biometrika*, 105(4):987–993, 2018.

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.

Mark R Montgomery, Michele Gragnolati, Kathleen A Burke, and Edmundo Paredes. Measuring living standards with proxy variables. *Demography*, 37(2):155–174, 2000.

Chris Paxton, Alexandru Niculescu-Mizil, and Suchi Saria. Developing predictive models using electronic medical records: challenges and pitfalls. In *AMIA Annual Symposium Proceedings*, volume 2013, page 1109. American Medical Informatics Association, 2013.

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

Judea Pearl. On measurement bias in causal inference. *arXiv preprint arXiv:1203.3504*, 2012.

Judea Pearl and Dana Mackenzie. *The Book of Why*. Allen Lane, 2018.

Jonas Peters, Dominik Janzing, and Bernhard Schölkopf. *Elements of causal inference: foundations and learning algorithms*. MIT press, 2017.

Niranjani Prasad, Li-Fang Cheng, Corey Chivers, Michael Draugelis, and Barbara E Engelhardt. A reinforcement learning approach to weaning of mechanical ventilation in intensive care units. *arXiv preprint arXiv:1704.06300*, 2017.

Aniruddh Raghu, Matthieu Komorowski, Imran Ahmed, Leo Celi, Peter Szolovits, and Marzyeh Ghassemi. Deep reinforcement learning for sepsis treatment. *arXiv preprint arXiv:1711.09602*, 2017a.

Aniruddh Raghu, Matthieu Komorowski, Leo Anthony Celi, Peter Szolovits, and Marzyeh Ghassemi. Continuous state-space models for optimal sepsis treatment-a deep reinforcement learning approach. *arXiv preprint arXiv:1705.08422*, 2017b.

Nathan Ramoly, Amel Bouzeghoub, and Beatrice Finance. A causal multi-armed bandit approach for domestic robots' failure avoidance. In *International Conference on Neural Information Processing*, pages 90–99. Springer, 2017.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082*, 2014.

Martin Riedmiller. Neural fitted q iteration–first experiences with a data efficient neural reinforcement learning method. In *European Conference on Machine Learning*, pages 317–328. Springer, 2005.

Suchi Saria, Daphne Koller, and Anna Penn. Learning individual and population level traits from clinical temporal data. In *Proceedings of Neural Information Processing Systems*, pages 1–9. Citeseer, 2010.

B. Schölkopf, D. Hogg, D. Wang, D. Foreman-Mackey, D. Janzing, C. J. Simon-Gabriel, and J. Peters. Modeling confounding by half-sibling regression. *Proceedings of the National Academy of Science*, 113(27):7391–7398, 2016. URL http://www.pnas.org/content/113/27/7391.full.

Peter Schulam and Suchi Saria. Reliable decision support using counterfactual models. In *Advances in Neural Information Processing Systems*, pages 1697–1708, 2017.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International Conference on Machine Learning*, pages 1889–1897, 2015.

Rajat Sen, Karthikeyan Shanmugam, Murat Kocaoglu, Alexandros G Dimakis, and Sanjay Shakkottai. Contextual bandits with latent confounders: An nmf approach. *arXiv preprint arXiv:1606.00119*, 2016.

Susan M Shortreed, Eric Laber, Daniel J Lizotte, T Scott Stroup, Joelle Pineau, and Susan A Murphy. Informing sequential clinical decision-making through reinforcement learning: an empirical study. *Machine learning*, 84(1-2):109–136, 2011.

Hossein Soleimani, Adarsh Subbaswamy, and Suchi Saria. Treatment-response models for counterfactual reasoning with continuous-time, continuous-valued interventions. *arXiv preprint arXiv:1704.02038*, 2017.

Richard S Sutton, Andrew G Barto, et al. *Reinforcement learning: An introduction*. MIT press, 1998.

Dustin Tran, Rajesh Ranganath, and David M Blei. The variational gaussian process. *arXiv preprint arXiv:1511.06499*, 2015.

Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *AAAI*, volume 2, page 5. Phoenix, AZ, 2016.

Jeffrey M Wooldridge. On estimating firm-level production functions using proxy variables to control for unobservables. *Economics Letters*, 104(3):112–114, 2009.

Wojciech Zaremba and Ilya Sutskever. Learning to execute. *arXiv preprint arXiv:1410.4615*, 2014.

Junzhe Zhang and Elias Bareinboim. Transfer learning in multi-armed bandit: a causal approach. In *Proceedings of the 16th Conference on Autonomous Agents and MultiA-gent Systems*, pages 1778–1780. International Foundation for Autonomous Agents and Multiagent Systems, 2017.

Appendices

A. Variational Lower Bound for $M_{\textbf{decon}}$

$$\log p_\theta(\vec{x}, \vec{a}, \vec{r})$$

$$= \log \int_u \int_{\vec{z}} p_\theta(\vec{x}, \vec{a}, \vec{r}, \vec{z}, u) \mathrm{d}\vec{z}\mathrm{d}u$$

$$\geq \int_u \int_{\vec{z}} q_\phi(\vec{z}, u | \vec{x}, \vec{a}, \vec{r}) \log \frac{p_\theta(\vec{x}, \vec{a}, \vec{r}, \vec{z}, u)}{q_\phi(\vec{z}, u | \vec{x}, \vec{a}, \vec{r})} \mathrm{d}\vec{z}\mathrm{d}u$$

$$= \int_u \int_{\vec{z}} q_\psi(u | \vec{x}, \vec{a}, \vec{r}) q_\phi(\vec{z} | \vec{x}, \vec{a}, \vec{r}) \log \frac{p_\theta(\vec{x}, \vec{a}, \vec{r}, \vec{z}, u)}{q_\psi(u | \vec{x}, \vec{a}, \vec{r}) q_\phi(\vec{z} | \vec{x}, \vec{a}, \vec{r})} \mathrm{d}\vec{z}\mathrm{d}u \qquad \text{(factorization assumption)}$$

$$= \int_u \int_{\vec{z}} q_\psi(u | \vec{x}, \vec{a}, \vec{r}) q_\phi(\vec{z} | \vec{x}, \vec{a}, \vec{r})$$

$$\log \frac{p(u)p(z_1)\left[\prod_{t=1}^{T} p(x_t|z_t, u)p(a_t|z_t, u)p(r_{t+1}|z_t, a_t, u)\right]\left[\prod_{t=2}^{T} p(z_t|z_{t-1}, a_{t-1})\right]}{q_\psi(u | \vec{x}, \vec{a}, \vec{r}) q_\phi(\vec{z} | \vec{x}, \vec{a}, \vec{r})} \mathrm{d}\vec{z}\mathrm{d}u$$

$$= \int_u \int_{\vec{z}} q(u | \vec{x}, \vec{a}, \vec{r}) q(z_1 | \vec{x}, \vec{a}, \vec{r}) \cdots q(z_T | z_{T-1}, \vec{x}, \vec{a}, \vec{r})$$

$$\log \frac{p(u)p(z_1)\prod_{t=1}^{T} p(x_t|z_t, u)p(a_t|z_t, u)p(r_{t+1}|z_t, a_t, u)\prod_{t=2}^{T} p(z_t|z_{t-1}, a_{t-1})}{q(u | \vec{x}, \vec{a}, \vec{r}) q(z_1 | \vec{x}, \vec{a}, \vec{r}) \cdots q(z_T | z_{T-1}, \vec{x}, \vec{a}, \vec{r})} \mathrm{d}\vec{z}\mathrm{d}u$$

$$= \sum_{t=1}^{T} \int_u \int_{z_1} \cdots \int_{z_T} q(u | \vec{x}, \vec{a}, \vec{r}) q(z_1 | \vec{x}, \vec{a}, \vec{r}) \cdots q(z_T | z_{T-1}, \vec{x}, \vec{a}, \vec{r})$$

$$\log \left(p(x_t|z_t, u)p(a_t|z_t, u)p(r_{t+1}|z_t, a_t, u) \right) \mathrm{d}\vec{z}\mathrm{d}u$$

$$+ \int_u \int_{z_1} \cdots \int_{z_T} q(u | \vec{x}, \vec{a}, \vec{r}) q(z_1 | \vec{x}, \vec{a}, \vec{r}) \cdots q(z_T | z_{T-1}, \vec{x}, \vec{a}, \vec{r}) \log \frac{p(u)}{q(u | \vec{x}, \vec{a}, \vec{r})} \mathrm{d}\vec{z}\mathrm{d}u$$

$$+ \int_u \int_{z_1} \cdots \int_{z_T} q(u | \vec{x}, \vec{a}, \vec{r}) q(z_1 | \vec{x}, \vec{a}, \vec{r}) \cdots q(z_T | z_{T-1}, \vec{x}, \vec{a}, \vec{r}) \log \frac{p(z_1)}{q(z_1 | \vec{x}, \vec{a}, \vec{r})} \mathrm{d}\vec{z}\mathrm{d}u$$

$$+ \sum_{t=2}^{T} \int_u \int_{z_1} \cdots \int_{z_T} q(u | \vec{x}, \vec{a}, \vec{r}) q(z_1 | \vec{x}, \vec{a}, \vec{r}) \cdots q(z_T | z_{T-1}, \vec{x}, \vec{a}, \vec{r}) \log \frac{p(z_t|z_{t-1}, a_{t-1})}{q(z_t|z_{t-1}, \vec{x}, \vec{a}, \vec{r})} \mathrm{d}\vec{z}\mathrm{d}u$$

$$= \sum_{t=1}^{T} \int_u \int_{z_t} q(u | \vec{x}, \vec{a}, \vec{r}) q(z_t|z_{t-1}, \vec{x}, \vec{a}, \vec{r}) \log \left(p(x_t|z_t, u)p(a_t|z_t, u)p(r_{t+1}|z_t, a_t, u) \right) \mathrm{d}z_t\mathrm{d}u$$

$$+ \int_u q(u | \vec{x}, \vec{a}, \vec{r}) \log \frac{p(u)}{q(u | \vec{x}, \vec{a}, \vec{r})} \mathrm{d}u$$

$$+ \int_{z_1} q(z_1 | \vec{x}, \vec{a}, \vec{r}) \log \frac{p(z_1)}{q(z_1 | \vec{x}, \vec{a}, \vec{r})} \mathrm{d}z_1$$

$$+ \sum_{t=2}^{T} \int_{z_{t-1}} \int_{z_t} q(z_t|z_{t-1}, \vec{x}, \vec{a}, \vec{r}) \log \frac{p(z_t|z_{t-1}, a_{t-1})}{q(z_t|z_{t-1}, \vec{x}, \vec{a}, \vec{r})} \mathrm{d}z_t\mathrm{d}z_{t-1}$$

$$
= \sum_{t=1}^{T} \mathop{\mathbb{E}}_{\substack{z_t \sim q(z_t | z_{t-1}, \vec{x}, \vec{a}, \vec{r}) \\ u \sim q(u | \vec{x}, \vec{a}, \vec{r})}} \left[\log p(x_t | z_t, u) + \log p(a_t | z_t, u) + \log p(r_{t+1} | z_t, a_t, u) \right]
$$

$$
- \mathrm{KL} \left(q(u | \vec{x}, \vec{a}, \vec{r}) || p(u) \right)
$$

$$
- \mathrm{KL} \left(q(z_1 | \vec{x}, \vec{a}, \vec{r}) || p(z_1) \right)
$$

$$
- \sum_{t=2}^{T} \mathop{\mathbb{E}}_{z_{t-1} \sim q(z_{t-1} | z_{t-2}, \vec{x}, \vec{a}, \vec{r})} \left[\mathrm{KL} \left(q(z_t | z_{t-1}, \vec{x}, \vec{a}, \vec{r}) || p(z_t | z_{t-1}, a_{t-1}) \right) \right].
$$

B. Variational Lower Bound for M_{alt}

$$\log p_\theta(\vec{x}, \vec{a}, \vec{r})$$

$$= \log \int_{\vec{z}} p_\theta(\vec{x}, \vec{a}, \vec{r}, \vec{z}) \mathrm{d}\vec{z}$$

$$\geq \int_{\vec{z}} q_\phi(\vec{z}|\vec{x}, \vec{a}, \vec{r}) \log \frac{p_\theta(\vec{x}, \vec{a}, \vec{r}, \vec{z})}{q_\phi(\vec{z}|\vec{x}, \vec{a}, \vec{r})} \mathrm{d}\vec{z}$$

$$= \int_{\vec{z}} q_\phi(\vec{z}|\vec{x}, \vec{a}, \vec{r}) \log \frac{p(z_1) \left[\prod_{t=1}^{T} p(x_t|z_t)p(a_t|z_t)p(r_{t+1}|z_t, a_t)\right] \left[\prod_{t=2}^{T} p(z_t|z_{t-1}, a_{t-1})\right]}{q_\phi(\vec{z}|\vec{x}, \vec{a}, \vec{r})} \mathrm{d}\vec{z}$$

$$= \int_{\vec{z}} q(z_1|\vec{x}, \vec{a}, \vec{r}) \cdots q(z_T|z_{T-1}, \vec{x}, \vec{a}, \vec{r})$$

$$\log \frac{p(z_1) \prod_{t=1}^{T} p(x_t|z_t)p(a_t|z_t)p(r_{t+1}|z_t, a_t) \prod_{t=2}^{T} p(z_t|z_{t-1}, a_{t-1})}{q(z_1|\vec{x}, \vec{a}, \vec{r}) \cdots q(z_T|z_{T-1}, \vec{x}, \vec{a}, \vec{r})} \mathrm{d}\vec{z}$$

$$= \sum_{t=1}^{T} \int_{z_1} \cdots \int_{z_T} q(z_1|\vec{x}, \vec{a}, \vec{r}) \cdots q(z_T|z_{T-1}, \vec{x}, \vec{a}, \vec{r}) \log \left(p(x_t|z_t)p(a_t|z_t)p(r_{t+1}|z_t, a_t)\right) \mathrm{d}\vec{z}$$

$$+ \int_{z_1} \cdots \int_{z_T} q(z_1|\vec{x}, \vec{a}, \vec{r}) \cdots q(z_T|z_{T-1}, \vec{x}, \vec{a}, \vec{r}) \log \frac{p(z_1)}{q(z_1|\vec{x}, \vec{a}, \vec{r})} \mathrm{d}\vec{z}$$

$$+ \sum_{t=2}^{T} \int_{z_1} \cdots \int_{z_T} q(z_1|\vec{x}, \vec{a}, \vec{r}) \cdots q(z_T|z_{T-1}, \vec{x}, \vec{a}, \vec{r}) \log \frac{p(z_t|z_{t-1}, a_{t-1})}{q(z_t|z_{t-1}, \vec{x}, \vec{a}, \vec{r})} \mathrm{d}\vec{z}$$

$$= \sum_{t=1}^{T} \int_{z_t} q(z_t|z_{t-1}, \vec{x}, \vec{a}, \vec{r}) \log \left(p(x_t|z_t)p(a_t|z_t)p(r_{t+1}|z_t, a_t)\right) \mathrm{d}z_t$$

$$+ \int_{z_1} q(z_1|\vec{x}, \vec{a}, \vec{r}) \log \frac{p(z_1)}{q(z_1|\vec{x}, \vec{a}, \vec{r})} \mathrm{d}z_1$$

$$+ \sum_{t=2}^{T} \int_{z_{t-1}} \int_{z_t} q(z_t|z_{t-1}, \vec{x}, \vec{a}, \vec{r}) \log \frac{p(z_t|z_{t-1}, a_{t-1})}{q(z_t|z_{t-1}, \vec{x}, \vec{a}, \vec{r})} \mathrm{d}z_t \mathrm{d}z_{t-1}$$

$$= \sum_{t=1}^{T} \mathbb{E}_{z_t \sim q(z_t|z_{t-1}, \vec{x}, \vec{a}, \vec{r})} \left[\log p(x_t|z_t) + \log p(a_t|z_t) + \log p(r_{t+1}|z_t, a_t)\right]$$

$$- \mathrm{KL}\left(q(z_1|\vec{x}, \vec{a}, \vec{r})||p(z_1)\right)$$

$$- \sum_{t=2}^{T} \mathbb{E}_{z_{t-1} \sim q(z_{t-1}|z_{t-2}, \vec{x}, \vec{a}, \vec{r})} \left[\mathrm{KL}\left(q(z_t|z_{t-1}, \vec{x}, \vec{a}, \vec{r})||p(z_t|z_{t-1}, a_{t-1})\right)\right].$$

C. Bi-directional LSTM

In our inference model, we use a similar architecture of bi-directional LSTM to that in (Krishnan et al., 2017). Apart from different inputs, the main difference is to introduce a_{t-1} to computing the combined hidden feature as shown in the following formula.
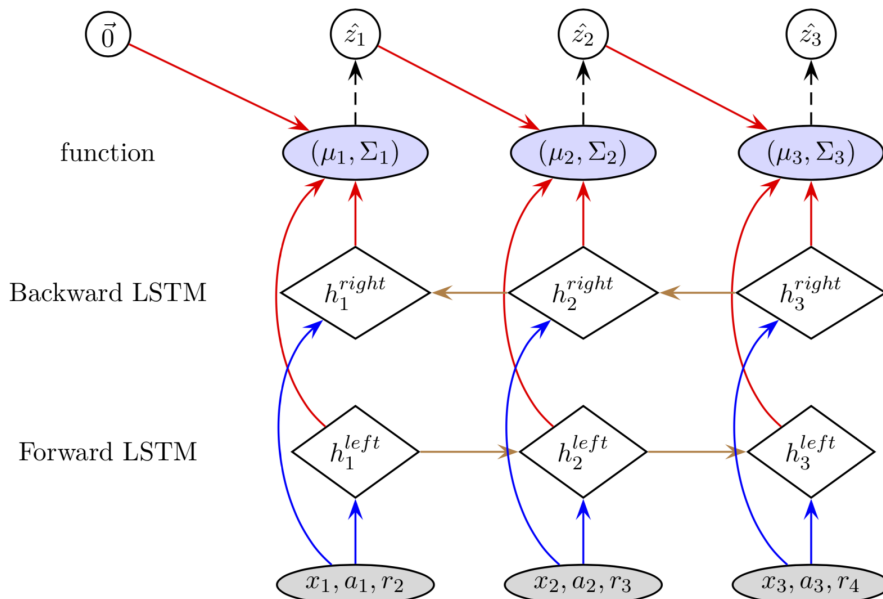
$$h_{\text{combined}} = \frac{1}{4}\left(tanh(W_z \cdot z_{t-1} + b_z) + tanh(W_a \cdot a_{t-1} + b_a) + h_t^{\text{left}} + h_t^{\text{right}}\right)$$

$$\mu_t = W_\mu \cdot h_{\text{combined}} + b_\mu$$

$$\sigma_t^2 = \text{softplus}\left(W_{\sigma^2} \cdot h_{\text{combined}} + b_{\sigma^2}\right)$$

D. Example of a proxy variable

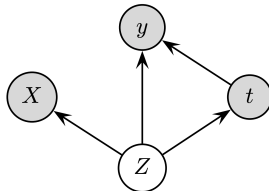For convenience, we directly move Figure 1 of (Louizos et al., 2017) here, as shown in Figure 7.

Figure 7: Example of a proxy variable. t is a treatment, e.g., medication; y is an outcome, e.g., mortality; Z is an unobserved confounder, e.g., socio-economic status; and X is noisy views on the hidden confounder Z, say income in the last year and place of residence.

E. Analysis of the Deconfounding Model

At each time step t, from the deconfounding model presented in Figure 2, we can extract two 4-tuple components, centering at z_t (Figure 8) and u (Figure 9), respectively. It is apparent to observe that both Figure 8 and Figure 9 share the exact same structure with Figure 7.
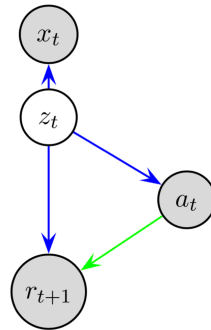

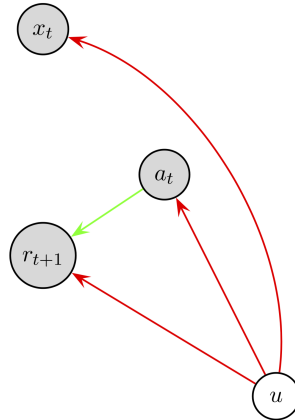Figure 8: The component of 4-tuple (z_t, x_t, a_t, r_{t+1}).

Figure 9: The component of 4-tuple (u, x_t, a_t, r_{t+1}).

F. Confounding Datasets

The probabilities between the confounder, action and reward are presented as follows,

$$p(u = 0) = 0.8, \quad p(u = 1) = 0.2; \tag{25}$$
$$p(a = T_1|u = 0) = 0.24, \quad p(a = T_2|u = 0) = 0.76; \tag{26}$$
$$p(a = T_1|u = 1) = 0.77, \quad p(a = T_2|u = 1) = 0.23; \tag{27}$$
$$p(r_c = R_1|a = T_1, u = 0) = 0.93, \quad p(r_c = R_2|a = T_1, u = 0) = 0.07; \tag{28}$$
$$p(r_c = R_1|a = T_2, u = 0) = 0.87, \quad p(r_c = R_2|a = T_2, u = 0) = 0.13; \tag{29}$$
$$p(r_c = R_1|a = T_1, u = 1) = 0.73, \quad p(r_c = R_2|a = T_1, u = 1) = 0.27; \tag{30}$$
$$p(r_c = R_1|a = T_2, u = 1) = 0.69, \quad p(r_c = R_2|a = T_2, u = 1) = 0.31; \tag{31}$$
$$\mu_{r_c=R_1} = -1, \quad \mu_{r_c=R_2} = -200, \quad \sigma = 2. \tag{32}$$

Note that, in our model r_c is a function of u, a_t, and z_t. We omit the notation of the state in the conditional probability of r_c because z_t is fixed at each time step, which does not affect the probability. To help readers understand this setting, we can make a straightforward analogy. More specifically, the confounder u stands for the socio-economic status where $u = 0$ represents the poor and $u = 1$ the rich. Treatment T_1 is more expensive but much better than Treatment T_2. R_1 means a healthier feedback/recovery than R_2. Equation (25) says that the poor are much more than the rich (it is indeed in reality). Equation (26) and Equation (27) tell us the fact that the poor tend to choose the cheaper but worse treatment T_2 whilst the rich prefer the more expensive but better treatment T_1, which also makes sense in the real world. Finally, Equations (28)-(31) reveal that T_1 is a better treatment than T_2 within both poor and rich population.

G. Experimental Settings

Hyperparameter	Value
Deconfounding Model	
learning rate	0.0001
dimension of z_t	50
dimension of x_t	784
dimension of a_t	1
dimension of r_t	1
dimension of u	2
dimension of LSTM unit	100
batch size	128
number of steps	5
number of epoch	400
Deconfounding AC	
number of episodes	1500
number of time steps in each episode	200
capacity of the replay memory	100,000
batch size	128
sample size of u	200

Table 1: Hyperparameters for deconfounding reinforcement learning. Note that for simplicity, in Section 4.3 the prior distribution over u is assumed to be a Bernoulli with parameter $p = 0.5$ and the dimension of u is set to 1.

Function	Architecture
	Deconfounding Model
f_1, f_2	$FC_{512} \to Conv_{7,32} \to Conv_{14,16} \to Conv_{28,1} \to FC_{784} \to \{\text{sigmoid, softplus}\}$
f_3, f_4	$\{FC_{100}, FC_{100}\} \to FC_{200} \to FC_{200} \to FC_1 \to \{\text{tanh, softplus}\}$
f_5, f_6	$\{FC_{100}, FC_{100}, FC_{100}\} \to FC_{100} \to FC_{100} \to FC_1 \to \{\text{sigmoid, softplus}\}$
f_7, f_8	$\{FC_{100}, FC_{100}\} \to FC_{100} \to FC_{100} \to FC_{50} \to \{\text{None, softplus}\}$
f_9, f_{10}	$\{[Conv_{5,16} \to Conv_{5,32} \to Conv_{5,32} \to FC_{100}], FC_{100}, FC_{100}\} \to FC_{100} \to FC_{100} \to LSTM_{100,5} \to \{\text{None, softplus}\}$
f_{11}, f_{12}	$\{[Conv_{5,16} \to Conv_{5,32} \to FC_{100}], FC_{100}, FC_{100}\} \to FC_{100} \to FC_{100} \to LSTM_{100,5} \to FC_{50} \to \{\text{None, softplus}\}$
f_{13}, f_{14}	$Conv_{5,16} \to Conv_{5,32} \to Conv_{5,32} \to FC_1 \to \{\text{tanh, softplus}\}$
f_{15}, f_{16}	$\{[Conv_{5,16} \to Conv_{5,32} \to Conv_{5,32} \to FC_{100}], FC_{100}\} \to FC_{100} \to FC_1 \to \{\text{sigmoid, softplus}\}$
	Deconfounding AC
$V(z_t; \phi_V)$	$FC_{300} \to FC_{300} \to \{\text{None, softplus}\}$
$\pi(a_t \mid z_t; \theta)$	$FC_{300} \to FC_{300} \to \{\text{tanh, softplus}\}$

Table 2: Architectures for deconfounding reinforcement learning. Here FC_k stands for a fully-connected layer with k units, $Conv_{k,n}$ for a convolution layer with n filters of size $k \times k$, $LSTM_{n,t}$ for a LSTM layer rolling out for t steps with latent size of n, $\{\cdot\}$ for the parallel operators, and $[\cdot]$ for the sequential operators. In default, FC_k and $Conv_{k,n}$ are followed by a softplus activation layer and a batch-norm layer, which are omitted here for simplicity. Note that, in our setting, the two functions in each pair $\{f_{2i-1}, f_{2i}\}_{i=1,\ldots,8}$ share the same parameters except for the last layer.