Jointly Learning Consistent Causal Abstractions Over Multiple Interventional Distributions

Fabio Massimo Zennaro FABIO.ZENNARO@WARWICK.AC.UK
Máté Drávucz MATE.DRAVUCZ@WARWICK.AC.UK
Geanina Apachitei GEANINA.APACHITEI@WARWICK.AC.UK
W. Dhammika Widanage DHAMMIKA.WIDANALAGE@WARWICK.AC.UK
Theodoros Damoulas T.DAMOULAS@WARWICK.AC.UK
Dept. of Computer Science & Dept. of Statistics & WMG, University of Warwick, Coventry, CV4 7AL, UK
The Faraday Institution, Harwell Science and Innovation, Campus, Quad One, Didcot, UK

Editors: Mihaela van der Schaar, Dominik Janzing and Cheng Zhang

Abstract

An abstraction can be used to relate two structural causal models representing the same system at different levels of resolution. Learning abstractions which guarantee consistency with respect to interventional distributions would allow one to jointly reason about evidence across multiple levels of granularity while respecting the underlying cause-effect relationships. In this paper, we introduce a first framework for causal abstraction learning between SCMs based on the formalization of abstraction recently proposed by Rischel (2020). Based on that, we propose a differentiable programming solution that jointly solves a number of combinatorial sub-problems, and we study its performance and benefits against independent and sequential approaches on synthetic settings and on a challenging real-world problem related to electric vehicle battery manufacturing.

Keywords: structural causal models, causal abstraction, causal representation learning

1. Introduction

Causal models are conceptual constructs we use in our everyday understanding of the world and in scientific modelling. Structural causal models (SCM) provide a mathematical formalism to express causal assumptions, encode quantities of interest, and reason about relationships of cause and effect. For instance, a research lab L may investigate lung cancer, and decide to model this scenario considering causal connections between a set of relevant variables, such as the smoking habits of patients, the presence of tar deposits in their lungs, and whether they ended up developing lung cancer (see Figure 1(left)). Another common feature of reasoning and scientific modelling is reliance on multiple levels of abstraction, whereby an identical system can be studied at multiple levels of granularity. For instance, in studying lung cancer, another research lab L' may decide to record only two variables, ignoring the contribution of tar deposits (see Figure 1(right)).

While SCMs allow us to deal with causal relationships internal to a given model, an abstraction focuses on *external* relations between different models. The idea of abstraction is widespread in artificial intelligence: it underlies the very notion of intelligence (Mitchell, 2021), it has been invoked to explain the success of deep learning (LeCun et al., 2015), and it has a central role in causal representation learning (Schölkopf et al., 2021). However, rigorous formalisms for abstractions have only been recently proposed (Rubenstein et al., 2017; Beckers and Halpern, 2019; Rischel, 2020).

© 2023 F.M. Zennaro, M. Drávucz, G. Apachitei, W.D. Widanage & T. Damoulas.

Our work starts from the abstraction framework of Rischel (2020), which provides a grounded way to express an abstraction between two SCMs and to quantitatively assess its consistency. Evaluation of consistency requires, beyond the definition of the SCMs, a full specification of an abstraction, which, in reality, may rarely be available. In this paper, we consider the problem of *learning an abstraction* when only partial information about it is available. This would correspond, for instance, to the case in which two labs are aware of their own SCMs but do not have an exact mapping between them. Successfully learning an abstraction would enable them to automatically transfer data and results across the models; in a low-data regime, where collecting new samples may be costly, having a proper abstraction would allow them to integrate their evidence and improve inferences.

Our contributions are introducing a new learning problem with specific semantics (abstraction learning) but a very generic syntax (commutativity learning); proposing a relaxation of the ensuing combinatorial problem and a solution based on differentiable programming which jointly solves a number of combinatorial sub-problems at once; analyzing empirically the performance of this approach on synthetic settings; demonstrating the benefits of our approach on the important problem of learning coating models for the batteries of electric vehicles (EV) by learning an abstraction that allows us to relate small-scale datasets collected through expensive real-world experiments performed in labs across France and the UK. To the best of our knowledge, this work also constitutes the first real-world application of an abstraction learning framework between SCMs.

The rest of the paper is organized as follows. Section 2 reviews important background definitions, and Section 3 presents related work. Section 4 discusses the learning problem. Section 5 introduces our proposed methodology, and Section 6 presents our empirical results. Section 7 summarizes our work and offers considerations on our approach and results. Appendix A offers a summary of the notation used throughout this paper.

Figure 1: Lung cancer SCM designed by lab L (left) and lab L' (right).

2. Background

In this section we provide definitions for the main concepts related to causality and abstraction.

2.1. Causality

In order to express causal models, we rely on Pearl's formalism of structural causal models (Pearl, 2009; Peters et al., 2017). See Pearl (2009) for the analogies and differences to the potential outcomes framework by Rubin (Rubin, 2005).

Definition 1 (Structural Causal Model (SCM) (Pearl, 2009)) *A structural causal model $\mathcal{M}$ consists of a directed acyclic graph (DAG) $\mathcal{G}_{\mathcal{M}}$, and a tuple $\langle \mathcal{U}, \mathcal{X}, \mathcal{F}, P(\mathcal{U}) \rangle$, where:*

- *$\mathcal{U}$ is a finite set of exogenous (unobserved or latent) variables;*
- *$\mathcal{X}$ is a finite set of endogenous (observed) variables, each one with a discrete domain $\mathcal{M}[X_i]$;*
- *$\mathcal{F} = \{f_1, ..., f_{|\mathcal{X}|}\}$ is a set of modular functions (mechanisms) such that $x_i = f_i(pa(X_i))$, where x_i is the value of an endogenous variable $X_i \in \mathcal{X}$ and $pa(X_i) \subseteq (\mathcal{U} \cup \mathcal{X}) \setminus X_i$.*
- *$P(\mathcal{U})$ is a joint distribution over the exogenous variables.*

Notice that the assumptions of acyclicity and joint distribution over the exogenous variables imply a *semi-Markovian SCM*; in this setting, the set $pa(X_i)$ of variables that determines the value of an endogenous variable X_i can be given the graph-theoretic interpretation of *parents* of X_i, and edges can be interpreted as causal relations. The additional assumption of discrete domains for each endogenous variable is introduced as a requirement of the abstraction framework of Rischel (2020). Models presented in Figure 1 can be given a rigorous reading as DAGs $\mathcal{G}_{\mathcal{M}}$ and $\mathcal{G}_{\mathcal{M}'}$.

A SCM allows us to study a system not only in the observational domain, but also under interventions. Modularity of the mechanisms allows us to formally define interventions.

Definition 2 (Intervention (Pearl, 2009)) *Given a SCM $\mathcal{M} = \langle \mathcal{U}, \mathcal{X}, \mathcal{F}, P(\mathcal{U}) \rangle$, an intervention $\iota := do(X_i = x_i)$ is an operator that generates a new post-interventional SCM $\mathcal{M}_\iota = \langle \mathcal{U}_\iota, \mathcal{X}_\iota, \mathcal{F}_\iota, P_\iota(\mathcal{U}_\iota) \rangle$ where $\mathcal{U}_\iota = \mathcal{U}$, $\mathcal{X}_\iota = \mathcal{X}$, $P_\iota(\mathcal{U}_\iota) = P(\mathcal{U})$, while $f_{\iota_i} = x_i$ and $f_{\iota_j} = f_j$ for all $j \neq i$.*

Thus, an intervention $\iota := do(X_i = x_i)$ creates a new model $\mathcal{M}_\iota$, identical to the original one except for the structural function f_i which is replaced with the constant x_i; the DAG of the post-interventional model $\mathcal{M}_\iota$ is similarly identical, with the exception of the node associated with X_i which has all the incoming edges removed. It is immediate to extend the definition of intervention to multiple endogenous variables: $do(\mathbf{X} = \mathbf{x})$, where $\mathbf{X} = [X_1, X_2, ..., X_m]$ is a vector of variables in $\mathcal{X}$ and $\mathbf{x} = [x_1, x_2, ..., x_m]$ is a vector of values associated with each variable.

2.2. Abstraction

We introduce a notion of abstraction meant to relate two SCMs representing an identical system. This definition originates from category theory, and it assumes SCMs with a finite set $\mathcal{X}$ of variables, each one defined on a finite and discrete domain $\mathcal{M}[X_i]$ (Rischel, 2020).

Definition 3 (Abstraction (Rischel, 2020)) *Given two SCMs $\mathcal{M}$ and $\mathcal{M}'$, an abstraction $\boldsymbol{\alpha}$ is a tuple $\langle R, a, \alpha \rangle$ where:*

- *$R \subseteq \mathcal{X}$ is a subset of relevant variables in the model $\mathcal{M}$;*
- *$a : R \to \mathcal{X}'$ is a surjective map between variables, from nodes in $\mathcal{M}$ to node in $\mathcal{M}'$;*
- *α is a collection of surjective maps $\alpha_{X'} : \mathcal{M}[a^{-1}(X')] \to \mathcal{M}'[X']$ from outcomes of variables in $\mathcal{M}$ to outcomes of variables in $\mathcal{M}'$.*

An abstraction establishes an asymmetric relation from a base or low-level model $\mathcal{M}$ to an abstracted or high-level model $\mathcal{M}'$. This definition encodes a mapping on two layers: on a structural or graphical level between the nodes of the DAGs via a, and on a distributional level via the maps $\alpha_{X'}$ (Zennaro, 2022). Since we are dealing with causal models, we require the SCMs to behave consistently wrt interventions.

Definition 4 (Consistency wrt an interventional distribution) *Let $\boldsymbol{\alpha}$ be an abstraction from $\mathcal{M}$ to $\mathcal{M}'$. Let $\mathbf{X}'$ and $\mathbf{Y}'$ be two disjoint subsets of variables in $\mathcal{X}'$. The abstraction $\boldsymbol{\alpha}$ is consistent wrt the interventional distribution $P'(\mathbf{Y}'|do(\mathbf{X}'))$ if the following diagram commutes:*

$$\begin{tikzcd}
\mathcal{M}[\mathrm{a}^{-1}(\mathbf{X}')] \arrow[r, "\mu"] \arrow[d, "\alpha_{\mathbf{X}'}"'] & \mathcal{M}[\mathrm{a}^{-1}(\mathbf{Y}')] \arrow[d, "\alpha_{\mathbf{Y}'}"] \\
\mathcal{M}'[\mathbf{X}'] \arrow[r, "\nu"'] & \mathcal{M}'[\mathbf{Y}']
\end{tikzcd}$$

that is:

$$\alpha_{\mathbf{Y}'} \circ \mu = \nu \circ \alpha_{\mathbf{X}'}, \tag{1}$$

where μ and ν are the interventional distributions $P(a^{-1}(\mathbf{Y}')|do(a^{-1}(\mathbf{X}')))$ and $P'((\mathbf{Y}')|do((\mathbf{X}')))$.

Intuitively, commutativity means that we would obtain equivalent interventional results in two different ways: (i) by intervening on the base model, obtaining the interventional distribution μ on the base model and then abstracting via $\alpha_{\mathbf{Y}'}$; or, (ii) by intervening on the base model, abstracting via $\alpha_{\mathbf{X}'}$ and then obtaining the interventional distribution ν on the abstracted model. Formally, commutativity has a category-theoretic meaning in the category of `FinStoch` where objects are sets and arrows are stochastic matrices (Fritz, 2020). A rigorous explanation is provided in Rischel (2020), but here it is worth remarking that, when working with SCMs with finite variables and finite domains, every discrete variable (e.g.: X') is associated with its domain set (e.g.: $\mathcal{M}'[X']$), discrete distributions (e.g.: $P'(Y'|do(X'))$) can be encoded in stochastic matrices (e.g.: μ), and deterministic abstractions (e.g.: $\alpha_{X'}$) can also be represented as binary stochastic matrices. Given this interpretation, the commutativity equality in Equation 1 boils down to an equality between matrix products. More details on this algebraic encoding are offered in Appendix B.

As abstractions normally introduce approximations and rarely guarantee perfect commutativity, it is useful to define an abstraction error wrt to interventions.

Definition 5 (Abstraction error wrt an interventional distribution (Rischel, 2020)) *Let $\boldsymbol{\alpha}$ be an abstraction from SCM $\mathcal{M}$ to SCM $\mathcal{M}'$. Let $\mathbf{X}'$ and $\mathbf{Y}'$ be two disjoint subsets of variables in $\mathcal{X}'$. The abstraction error $E_{\boldsymbol{\alpha}}(\mathbf{X}', \mathbf{Y}')$ wrt the interventional distribution $P'(\mathbf{Y}'|do(\mathbf{X}'))$ is the Jensen-Shannon distance (JSD) between the upper and the lower path in the diagram in Definition 4:*

$$D_{JSD}(\alpha_{\mathbf{Y}'} \circ \mu; \nu \circ \alpha_{\mathbf{X}'}). \tag{2}$$

A definition of JSD is recalled in Appendix C. Intuitively, the abstraction error quantifies the discrepancy between the upper and lower path in the abstraction diagram: how different are the results when (i) we first work with the low-level model and then abstract, and (ii) we first abstract and then work with the high-level model. This measure and the choice of JSD have a category-theoretic justification in the category `FinStoch` enriched in the category `Met` (Rischel, 2020).

From the above definitions, we can derive an overall notion of error.

Definition 6 (Abstraction error) *Let $\boldsymbol{\alpha}$ be an abstraction from $\mathcal{M}$ to $\mathcal{M}'$. Let $\mathcal{J}$ be the set of all disjoint pair sets $(\mathbf{X}', \mathbf{Y}') \in \mathscr{P}(\mathcal{X}') \times \mathscr{P}(\mathcal{X}'), \mathbf{X}' \cap \mathbf{Y}' = \emptyset$, where $\mathscr{P}()$ is the powerset. The abstraction error is:*

$$e(\boldsymbol{\alpha}) = \sup_{(\mathbf{X}', \mathbf{Y}') \in \mathcal{J}} E_{\boldsymbol{\alpha}}(\mathbf{X}', \mathbf{Y}').$$

Thus, the overall abstraction error is simply the worst-case abstraction error when considering all possible interventional distributions. While $\mathcal{J}$ is formally defined as the set of all disjoint pair

sets, it is possible to reduce $\mathcal{J}$ only to those pair sets $(\mathbf{X}', \mathbf{Y}')$ representing meaningful or relevant interventions $P'(\mathbf{Y}'|do(\mathbf{X}'))$. A consistent abstraction is then simply a zero-error abstraction:

Definition 7 (Consistent abstraction) *Let $\boldsymbol{\alpha}$ be an abstraction from $\mathcal{M}$ to $\mathcal{M}'$. The abstraction is consistent if, for all pairs $(\mathbf{X}', \mathbf{Y}')$ in $\mathcal{J}$, the abstraction $\boldsymbol{\alpha}$ is consistent wrt $P'(\mathbf{Y}'|do(\mathbf{X}'))$.*

3. Related Work

Alternative accounts of abstraction have been offered in the literature. A seminal definition proposed by Rubenstein et al. (2017), and refined by Beckers and Halpern (2019), characterized abstraction only on a distributional level; measures of abstraction approximation in this context are discussed in Beckers et al. (2020). More detailed definitions that consider both the structural and the distributional levels, and that are grounded in category theory, have been presented in Rischel (2020); Rischel and Weichwald (2021); Otsuka and Saigo (2022). A review and a comparison of these definitions is offered in Zennaro (2022). An attempt at defining a hierarchy of abstraction learning problems has been put forward in Zennaro et al. (2022).

Causal representation learning (CRL) (Chalupka et al., 2017) shares with this work a similar objective, but a different setup. Instead of learning a mapping between two SCMs, CRL normally starts from unstructured data and aims at learning causally-relevant representations. While abstraction learning as we defined it deals with mappings between SCMs, CRL may be seen as a preliminary or complementary step to abstraction learning, dealing with a mapping from an unstructured data space onto the space of causal variables potentially belonging to a SCM.

4. Problem Statement

The abstraction framework in Section 2 provides a rigorous way to estimate an abstraction error once we are given a fully defined abstraction $\boldsymbol{\alpha} = \langle R, a, \alpha \rangle$ from $\mathcal{M}$ to $\mathcal{M}'$. Instead, we consider the scenario in which two SCMs $\mathcal{M}$ and $\mathcal{M}'$ are known, but only a partially defined abstraction in the form $\boldsymbol{\alpha} = \langle R, a \rangle$ is available. This represents the common situation where structural knowledge (R, a) is readily available, but detailed distributional knowledge (α) lacking; in our lab example, this corresponds to the case where researchers from lab L and L' can exchange their models, agree on which variables are relevant, but they have no immediate knowledge on how to relate the results of their interventional experiments.

Given two SMCs $\mathcal{M}$ to $\mathcal{M}'$, and a partial abstraction $\boldsymbol{\alpha} = \langle R, a \rangle$, abstraction learning is the problem of learning the values for the maps $\alpha_{X'}$ that minimize the abstraction error, achieving, if possible, a consistent abstraction. We can then cast the abstraction learning problem as an optimization problem:

$$\min_{\alpha \in \mathcal{A}} e(\boldsymbol{\alpha}), \tag{3}$$

where $\mathcal{A}$ is the space of all feasible solutions for the collection of surjective maps $\alpha_{X'}$. This optimization problem is challenging for three reasons: (i) it implies multiple sub-problems; (ii) these sub-problems are related; (iii) the solution space is combinatorial.

Multiple sub-problems. Let us consider and unpack the optimization in Equation 3:

$$\min_{\alpha\in\mathcal{A}} \quad e(\boldsymbol{\alpha}) = \tag{4}$$

$$\min_{\alpha\in\mathcal{A}} \quad \sup_{(\mathbf{X}',\mathbf{Y}')\in\mathcal{J}} E_{\boldsymbol{\alpha}}(\mathbf{X}',\mathbf{Y}') = \tag{5}$$

$$\min_{\alpha\in\mathcal{A}} \quad \sup\left\{E_{\boldsymbol{\alpha}}(\mathbf{X}',\mathbf{Y}'), E_{\boldsymbol{\alpha}}(\mathbf{Y}',\mathbf{Z}'), E_{\boldsymbol{\alpha}}(\mathbf{X}',\mathbf{Z}')...\right\} = \tag{6}$$

$$\min_{\alpha\in\mathcal{A}} \quad \sup\left\{D_{JSD}(\alpha_{\mathbf{Y}'}\mu;\nu\alpha_{\mathbf{X}'}), D_{JSD}(\alpha_{\mathbf{Z}'}\mu';\nu'\alpha_{\mathbf{Y}'}), D_{JSD}(\alpha_{\mathbf{Z}'}\mu'';\nu''\alpha_{\mathbf{X}'})...\right\}. \tag{7}$$

The minimization of a supremum implies a minimization over multiple *sub-problems*. Each sub-problem is defined by a pair set of endogenous variables $(\mathbf{X}',\mathbf{Y}') \in \mathcal{J}$ representing the interventional distribution $P'(\mathbf{Y}'|do(\mathbf{X}'))$. For each interventional distribution we set up a diagram as in Definition 4, and we solve it in $\alpha_{\mathbf{X}'}$ and $\alpha_{\mathbf{Y}'}$ with the objective of minimizing the error $E_{\boldsymbol{\alpha}}(\mathbf{X}',\mathbf{Y}')$. Thus, we have a number of sub-problems equal to the cardinality $|\mathcal{J}|$, each one requiring the minimization of a JSD, as shown in Equation 7. Notice the direct correspondence between *one* sub-problem, *one* diagram, and *one* minimization of a JSD.

Related sub-problems. The sub-problems identified above are not necessarily independent. As soon as we consider two interventional distributions involving one identical subset of variables, we will obtain two diagrams sharing an abstraction variable, For instance, if we consider $P'(Y'|do(X'))$ and $P'(Z'|do(X'))$, then the induced diagrams will share the abstraction map $\alpha_{X'}$; this implies that minimizing the abstraction error for $E_{\boldsymbol{\alpha}}(X',Y')$ by changing the value of $\alpha_{X'}$ will affect the abstraction error of $E_{\boldsymbol{\alpha}}(X',Z')$ too.

Combinatorial optimization. The domain of each $\alpha_{X'}$, encoded as a binary stochastic matrix with shape $N_i \times M_i$, is $\{0,1\}^{N_i\times M_i}$, together with the constraint of stochasticity and surjectivity. There exists a finite number of solutions, equal to all possible surjective functions from a discrete M_i-dimensional space to a discrete N_i-dimensional space, which can be computed as $N!\left\{ {M \atop N} \right\}$, where $\left\{ {M \atop N} \right\} = \frac{1}{N!}\sum_{i=0}^{N}(-1)^{N-i}\binom{N}{i}i^M$ is the second-kind Stirling number (Graham et al., 1994). Consequently, since α is a collection of $|\mathcal{X}'|$ surjective functions, the space $\mathcal{A}$ of feasible solutions is the Cartesian product $\mathcal{A} = \prod_{i=1}^{|\mathcal{X}'|}\{0,1\}^{N_i\times M_i}$, with the constraint of stochasticity and surjectivity. The number of solutions, given by all possible combination of $\alpha_{X'}$ matrices, is:

$$|\mathcal{A}| = \prod_{i=1}^{|\mathcal{X}'|} N_i! \left\{ {M_i \atop N_i} \right\}, \tag{8}$$

The finiteness of the space $\mathcal{A}$ allows, theoretically, for searching an optimal solution by *enumeration*. However, this quickly becomes unfeasible as the dimensionality of the input models grows.

5. Methodology

As the enumeration strategy is not generally feasible, some form of heuristic becomes necessary. In this section we first describe our solution approach to the abstraction learning problem discussing: (i) a joint approach to solve all the abstraction sub-problems; (ii) a relaxation of the optimization problem; (iii) a parametrization of the relaxed problem; (iv) the enforcement of stochasticity on the parameters; (v) the enforcement of surjectivity on the parameters; (vi) solution by gradient descent. We then present two immediate extensions of the algorithm: (vii) use of weighting on the loss function; and (viii) ensembling of models to better explore the solution space.

(i) Joint approach. The abstraction learning problem consists of a set of related sub-problems. Simplistic approaches could ignore the structure connecting these problems. An *independent approach*, for instance, would consider each problem separately, solve it, and, at the end, apply some form of aggregation to decide the value of those matrices appearing in multiple sub-problems. Similarly, a *sequential approach* would establish an order among the sub-problems, and solve them accordingly, freezing the value of previously learned matrices. However, we aim at devising a *joint approach* which, from the beginning, would take into considerations the existing structure and compute a jointly optimal solution over all the sub-problems at once. We will compare our joint approach against these simpler approaches and showcase the benefits of our algorithm.

(ii) Problem relaxation. Traversing the discrete solution space implied by a combinatorial optimization problem is notoriously challenging (Papadimitriou and Steiglitz, 1998) and the subject of current research (Titsias and Yau, 2017; Jaini et al., 2021). We introduce a relaxation of the original problem, by redefining the original solution space $\mathcal{A}$ as $\mathcal{A}' = \prod_{i=1}^{|\mathcal{X}|}[0,1]^{N_i \times M_i}$. Solution matrices are now allowed to assume continuous values in $[0,1]$, which can be interpreted as allowing for stochastic or uncertain mappings.

(iii) Parametrization of the problem. Having allowed for a continuous solution space, we now parametrize the problem by defining a set of parameters $\mathbf{W}$ containing a weight matrix $W_{X'} \in \mathbb{R}^{N_i \times M_i}$ for each abstraction $\alpha_{X'}$. We then redefine our learning problem as:

$$\min_{\mathbf{W} \in \mathcal{A}''} e(\boldsymbol{\alpha}(\mathbf{W})), \tag{9}$$

where the abstraction tuple $\boldsymbol{\alpha}(\mathbf{W}) = \langle R, a, \mathbf{W} \rangle$ is now parametrized by $\mathbf{W}$, and the solution space is $\mathcal{A}'' = \prod_{i=1}^{k} \mathbb{R}^{N_i \times M_i}$.

(iv) Enforcement of stochasticity. The solution to the parametrized problem in Equation 9 returns a set of weights matrices in $\mathbb{R}^{N_i \times M_i}$. To force the solution towards a binary form, we apply a tempering operator column-wise $t_{col}(W) = \frac{e^{\frac{W_{i,j}}{T}}}{\sum_i e^{\frac{W_{i,j}}{T}}}$, where $T \in \mathbb{R}_+$ is the temperature hyperparameter. This operator projects the solution into $[0,1]^{N_i \times M_i}$, approaching a binary solution in $\{0,1\}^{N_i \times M_i}$ as $T \to 0$. The problem can be re-expressed as:

$$\min_{\mathbf{W} \in \mathcal{A}''} \mathcal{L}_1(\mathbf{W}) = \min_{\mathbf{W} \in \mathcal{A}''} e(\boldsymbol{\alpha}(t_{col}(\mathbf{W}))). \tag{10}$$

Notice how the relaxation and the parametrization allowed us to move from the solution space $\mathcal{A}$ to $\mathcal{A}'$ to $\mathcal{A}''$, and how the tempering operator has allowed us to approximately project back to $\mathcal{A}$.

(v) Enforcement of surjectivity. While the tempering operator returns a solution with the desired binary form, nothing guarantees that the solution matrices $t_{col}(W_{X'})$ will encode surjective functions. To enforce this property we introduce a second term in our loss function which penalizes every row in the learned matrices which do not contain at least a single one:

$$\min_{\mathbf{W} \in \mathcal{A}''} \mathcal{L}_2(\mathbf{W}) = \min_{\mathbf{W} \in \mathcal{A}''} \sum_{W \in \mathbf{W}} \sum_{i=1}^{N_i} \left(1 - \max_j t_{col}(W)_{ij}\right). \tag{11}$$

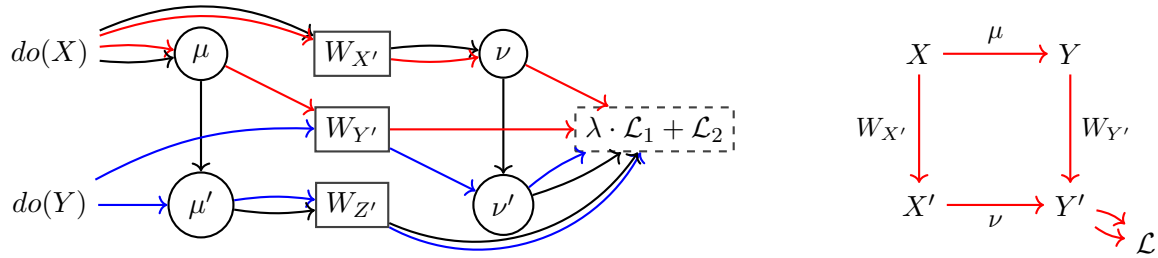


Figure 2: Neural network structure implied by the joint approach on an abstraction learning problem with $\mathcal{J} = \{(X', Y'), (X', Z'), (Y', Z')\}$ (left). Circles represent known interventional distributions, solid rectangles learnable parameters, and the dashed box the loss function. Colors trace the diagrams defined by (X', Y') (red), (X', Z') (black), and (Y', T') (blue); following, for instance, the red lines, it is possible reconstruct the upper and lower path as in the diagram of (X', Y') (right). The network structure highlights the possibility of learning at once, via backpropagation, all the weight matrices shared by multiple diagrams. Contrast with other approaches in Appendix F.

(vi) Solution by gradient descent. Let our loss function be:

$$\min_{\mathbf{W} \in \mathcal{A}''} \mathcal{L}(\mathbf{W}) = \min_{\mathbf{W} \in \mathcal{A}''} \lambda \mathcal{L}_1(\mathbf{W}) + \mathcal{L}_2(\mathbf{W}), \tag{12}$$

where $\lambda \in \mathbb{R}_+$ is a trade-off hyperparameter. $\mathcal{L}(\mathbf{W})$ is given by the sum of piecewise continuous functions: the first term is the supremum of JSDs, while the second term is related to the sum of maxima in the rows of the parameters. Given a random starting set of candidate solutions $\mathbf{W}$, it is possible to move in the solution space towards a locally optimal solution via gradient descent algorithms, iteratively computing $\mathbf{W} = \mathbf{W} - \eta \frac{\partial \mathcal{L}(\mathbf{W})}{\partial \mathbf{W}}$, with $\eta \in \mathbb{R}_+$ being a learning rate.

Algorithm 1 in Appendix E.1 summarizes our joint approach. By relying on automatic differentiation we can see the algorithm as a form of neural network encoding the solution in the weights and having weight sharing defined by $\mathcal{J}$, as shown in Figure 2.

(vii) Weighting by interventional data. The loss term in Equation 10 implicitly weights each abstraction error $E_{\boldsymbol{\alpha}}(\mathbf{X}', \mathbf{Y}')$ uniformly. It may be desirable, however, to scale the error with the relevance of the different interventional distributions using a vector $\boldsymbol{\kappa} \in \mathbb{R}_+^{|\mathcal{J}|}$:

$$\min_{\mathbf{W} \in \mathcal{A}''} \quad \mathcal{L}_1(\mathbf{W}, \boldsymbol{\kappa}) = \min_{\mathbf{W} \in \mathcal{A}''} e(\boldsymbol{\alpha}(t_{col}(\mathbf{W}), \boldsymbol{\kappa})) = \tag{13}$$

$$\min_{\mathbf{W} \in \mathcal{A}''} \quad \sup \left\{ \kappa_1 D_{JSD}(\alpha_{\mathbf{Y}'} \mu; \nu \alpha_{\mathbf{X}'}), \kappa_2 D_{JSD}(\alpha_{\mathbf{Z}'} \mu'; \nu' \alpha_{\mathbf{Y}'}) ... \right\}. \tag{14}$$

Assuming that more relevant interventions are collected more often, these weights $\boldsymbol{\kappa}$ may be set proportionally to the amount of interventions collected. Alternatively, importance schemes reweighting samples or interventions (Xu et al., 2021) or affecting the balance between JSD losses (Vandenhende et al., 2021) may be adopted.

(viii) Ensembling of models. Solving a relaxed combinatorial optimization problem by gradient descent does guarantee only the achievement of a local optimum (Papadimitriou and Steiglitz, 1998). In particular, gradient descent is sensitive to the morphology of the loss landscape and parameter initialization. Ensembling has been shown to improve the performance and the uncertainty

Table 1: Summary of the scenarios. *Scenario* and *Type of abstraction* describe the scenario; *Sub-pr.* refers to the number of sub-problems (and hence the number of diagrams) implied by each abstraction learning problem; *Abstraction Matrices* lists the shapes of the abstraction matrices to be learned; *#Sol.* evaluates the number of possible solutions according to Equation 8; *Optimum* reports the existence of a unique zero-error or non-zero error solution.

Scenario	***Type of abstraction***	***Sub-pr.***	***Abstraction Matrices***	***#Sol.***	***Optimum***
Basic	Elimination of low-level var	1	$\{2 \times 2, 2 \times 2\}$	4	Unique zero-error
Collapsing	Merging of low-level vars	3	$\{2 \times 4, 2 \times 2, 2 \times 2\}$	56	Unique non-zero-error
Extended	Reduction of resolution	3	$\{3 \times 4, 2 \times 3, 2 \times 2\}$	432	Unique zero-error
v-Structure	Elimination of low-level var Merging of low-level vars Reduction of resolution	3	$\{2 \times 2, 2 \times 2, 2 \times 4\}$	56	Unique non-zero-error

estimation of learning (Dietterich, 2000; Lakshminarayanan et al., 2017); we then rely on running an ensemble of models with different starting parameters $\mathbf{W}$ in order to learn better abstractions.

6. Experiments

In this section we report results from running our algorithm both on synthetic and real-world data. Data and code for all simulations are openly available at https://github.com/mattdravucz/jointly-learning-causal-abstraction/.

6.1. Synthetic experiments

In our synthetic simulations, we consider four scenarios featuring different aspects of abstraction and presenting different challenges. Table 1 provides an overview of these scenarios; as evinced from it, our scenarios cover different forms of abstractions, while keeping the complexity of the problems limited for verification via enumeration. Appendix G.1 provides details for all the models, the abstractions, the set $\mathcal{J}$, and the optimal solutions computed via enumeration.

For each one of these scenarios, we first compute the ground-truth solution via enumeration. Then we perform the following studies: (a) we run our algorithm, and compare its performance against an independent and a sequential baseline approach (see Section 5(i)); (b) we analyze how weighting can affect the quality of the solutions; (c) we perform an ablation study in which we remove the surjectivity penalty term; (d) we perform a sensitivity analysis in which we vary the value of the hyperparameters T and λ specified by our approach. Performance is evaluated in terms of JSD, surjective penalty, ℓ_1-distance from the optimal ground-truth solution, and wall-clock time. Results are averaged over 10 simulations. Algorithms for all the approaches are provided in Appendix E and details about the experimental settings and metrics in Appendix G.2.

Figure 3 exemplifies the learning process in the *extended* scenario. The joint approach reliably learns a solution closer to the ground-truth optimum than the other approaches. As low levels of JSD and surjective penalty are necessary but not sufficient to reach the optimal ground-truth solution, the independent and sequential approach achieve a low loss, but their ℓ_1-distance is significantly higher than the joint approach. We hypothesize that the better results of our algorithm are due to the reliance on the information shared between sub-problems that is discarded by the other approaches. Analogous plots for the other scenarios are available in Appendix G.3. Table 2 provides the perfor-

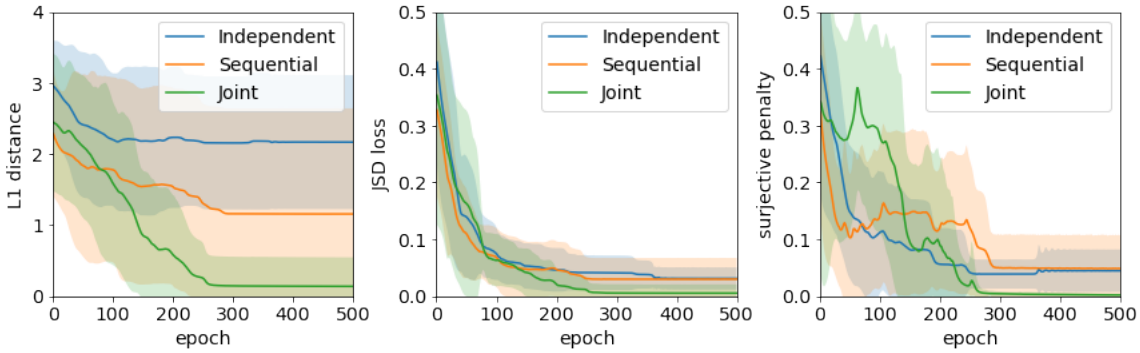


Figure 3: Performance during training on the *extended* scenario. While all approaches minimize their loss, the joint approach achieves a significantly lower (Wilcoxon test, p-value < 0.05) ℓ_1-distance from the ground truth.

Table 2: Performance at the end of training aggregated over the four scenarios. In all instances, the joint approach competes or overcomes the baseline approaches, although with longer runtime.

	L1 Dist	***JSD Loss***	***Surj Pen***	***Time***
Independent	3.25 ± 2.06	0.50 ± 0.31	0.40 ± 0.37	3.74 ± 0.05
Sequential	1.45 ± 1.55	0.28 ± 0.09	0.03 ± 0.07	7.10 ± 0.41
Joint	0.85 ± 0.64	0.24 ± 0.03	0.08 ± 0.11	8.12 ± 0.20

mance of the three approaches aggregated over the four scenarios. Consistently with our hypothesis, the joint approach performs better or equally to the other algorithms in terms of ℓ_1-distance from the ground truth by exploiting all the shared information; next comes the sequential approach which uses shared information only partially; last is the independent approach which completely discards it. For an analogous reason, ordering in terms of time efficiency is reversed: dropping shared information, the independent approach is fully parallelizable wrt the sub-problems; the sequential approach may deem some sub-problems redundant and avoid solving them; the joint approach considers all the sub-problems jointly at once. Table 3 shows how the final result of the joint algorithm may be affected by weighting. In this instance, we considered only the *v-structure* scenario, and we assumed that the JSD error related to cancer when intervening on smoking ($E_{\boldsymbol{\alpha}}(S', C')$) would be more important than the JSD error for fatigue when intervening on cancer ($E_{\boldsymbol{\alpha}}(C', F')$) or when intervening on smoking ($E_{\boldsymbol{\alpha}}(S', F')$). When placing 80% of the weight on $E_{\boldsymbol{\alpha}}(S', C')$, the final JSD for this interventional diagram decreases, while other JSD do not significantly change and exhibit higher variance. Weighting can then be used to get better approximations on those parts of the problem the modeller is more concerned with. Table 4 confirms the the critical role of the surjective penalty $\mathcal{L}_2$; in its absence the algorithm can learn a better solution in terms of JSD by ignoring values in the abstracted model, but it lands on a solution significantly further from the optimal solution. Additional discussion and sample learned matrices are provided in Appendix G.3, together with results from the sensitivity analysis.

Table 3: JSD on the *v-structure* scenario. Weighting causes a marginal redistribution of errors.

	$E_{\boldsymbol{\alpha}}(S', C')$	$E_{\boldsymbol{\alpha}}(C', F')$	$E_{\boldsymbol{\alpha}}(S', F')$
Unweighted	0.28 ± 0.07	0.28 ± 0.01	0.15 ± 0.03
Weighted	0.25 ± 0.05	0.27 ± 0.06	0.15 ± 0.04

Table 4: Performance on the *v-structure* scenario. Surjectivity penalty is critical to achieve significantly better (Wilcoxon test, p-value < 0.05) results.

	L1 Dist	***JSD Loss***
Joint	$\mathbf{3.00 \pm 1.34}$	0.72 ± 0.08
Ablated joint	5.00 ± 2.41	0.68 ± 0.15

6.2. Real-world experiment

We learn an abstraction between the implicit causal models for lithium-ion battery manufacturing developed by two research groups: the Laboratoire de Réactivité et Chimie des Solides (LRCS) group, and the Warwick Manufacturing Group (WMG). Battery electrode manufacturing is a complex process involving several key stages (e.g. material selection, mixing and coating.To develop high-performing batteries, it is necessary to understand how each of the manufacturing parameters influence the subsequent product. The current approach relies heavily on experienced lab personnel with extensive knowledge in order to adjust the manufacturing parameters and achieve a desired battery performance. Considerable research effort is directed to develop models of the manufacturing stages (Román-Ramírez et al., 2022; Niri et al., 2022) and optimised feedback control mechanisms to reduce the reliance on human expertise.

In this work we focus on the dry mass loading (ML) variable from the coating stage (Cunha et al., 2020; Liu et al., 2022), which directly determines the energy density of the final battery. The dry ML is in part controlled by the comma-bar gap (CG) variable, which is set manually and guides the mass loading of the active material in its wet form; the wet coating subsequently passes through a drying stage resulting in the dry coating. Public datasets on this process are scarce and of limited dimensions, due to the cost and complexity of the measurements; however, the ability to predict the dry ML based on the upstream CG variable is vital in order to achieve a target battery energy density and increase manufacturing efficiency. To obviate this problem, we aim at learning an abstraction that may relate the models assumed by the two research groups and then integrate their data to significantly improve downstream inferences.

We use a dataset for battery coating from the LRCS group (Cunha et al., 2020), and recordings performed by the WMG group. We perform pre-processing in order to select the relevant variables. As in many real-world scenarios, fully-specified SCMs are not available, so we define elementary SCMs $\mathcal{M}^{WMG}$ and $\mathcal{M}^{LRCS}$ with minimal assumptions (Figure 10 and 11). We then elect the WMG model as the base model since it has a higher resolution in terms of number and domain of observed variables. Appendix H.1 provides details about the data, pre-processing, the SCMs and the abstraction. We learn the abstraction $\boldsymbol{\alpha}$ from $\mathcal{M}^{WMG}$ to $\mathcal{M}^{LRCS}$ using our joint approach, similarly to what we have done with the synthetic experiments. We evaluate results in two ways. Qualitatively, we assess the shape of the learned matrices, to confirm they are binary and surjective, and to check whether identical values in the domain of the base and abstracted variables are mapped to each other. Quantitatively, we set three downstream regression problems, see Table 5, to assess whether transporting data via abstraction could help improve prediction. Task (a) represents a challenging scenario in which we use limited and expensive experimental data to perform predictions. Task (b) represents a scenario in which data transported via abstraction provides support for our

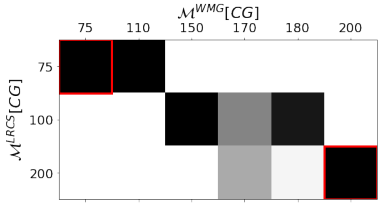


Figure 4: Learned W_{CG}. The matrix is close to binary, surjective, and diagonal as expected.

Table 5: Regression problems setup and MSE. Transporting data via abstraction reduces the error.

	Training set	***Test Set***	***MSE***
(a)	LRCS[$CG \neq k$]	LRCS[$CG = k$]	1.86 ± 1.75
(b)	LRCS[$CG \neq k$] + WMG	LRCS[$CG = k$]	0.22 ± 0.26
(c)	LRCS[$CG \neq k$] + WMG[$CG \neq k$]	LRCS[$CG = k$] + WMG[$CG = k$]	1.22 ± 0.95

predictions, while task (c) the case in which transported data augments the training set but does not provide support on the test set. See Appendix H.2 for details on experimental settings and metrics.

Figure 4 shows the learned matrix W_{CG} corresponding to the abstraction map α_{CG} relating the CG variables in the two models. The matrix approximates a binary matrix encoding a surjective function. A red border is used to denote identical value of the CG matrix in the base and abstracted model; these values are successfully mapped to each other, while intermediate values align along the main diagonal. Notice, however, that alternative patterns could also be learned, as discussed in Appendix H.3, together with shape of the learned matrix W_{ML}.

Table 5 shows the mean-square error when learning only on the small LRCS dataset (task a) and when using data transported via abstraction from the WMG dataset; performance improves both when WMG data provides the missing support for prediction (task b) or when it just augments the dataset (task c).; when not providing the required missing support, we observe an improvement on selected cases; this is likely due to having learned a non-perfect abstraction and to the noise introduced during the transport of the data.

7. Discussion and Conclusion

In this paper we have built an abstraction learning framework starting from the formalization of an abstraction between SCMs proposed in (Rischel, 2020). We have examined how we could use this definition to express the learning problem as an abstraction error minimization problem, and proposed an algorithm to solve it. Our approach has been based on relaxing the initial problem, parametrizing it and solving it via gradient descent. Results on synthetic datasets show that our algorithm perform better than other simpler approaches, and it reaches a solution closer to the optimum. Furthermore, application to EV manufacturing data provided a proof of concept of the usefulness of learning abstraction to relate models and transport data in low-data regimes.

The abstraction learning problem we have tackled is a particular instance of the very generic problem of learning a mapping not just between isolated sets or objects, but between sets and objects sharing a structure expressed by transformations or morphisms. This problem has been given the form of a notoriously challenging discrete combinatorial optimization problem, for which other common heuristics exist in the optimization literature. Application of our approach to the real-world data of a lithium-ion battery manufacturing stage provided a proof of concept of the potential of learning abstraction between models devised by different groups, although more research would be needed to make the proposed algorithm widely usable in real-world applications. The joint

approach is sensitive to initialization, and using abstraction to transfer data may introduce noise; increasing robustness and extending it to the continuous domain are directions for future work.

As a first instance of an abstraction learning problem, our setup assumes perfect knowledge of the models. Dropping this requirement leads to problems in which we could learn both abstraction and distributions from data (incidentally, in our real-world scenario, we trivially learned mechanisms from data at setup time, but the learning of mechanisms and abstraction could happen jointly). Similarly, we may further limit knowledge about the abstraction, dropping, for instance, the requirement of knowing a or R (Zennaro et al., 2022). Alternatively, it may also be possible to consider using domain knowledge on how specific variables and outputs are related to guide learning; the space of surjective function grows very rapidly, and restricting it by using available priors would simplify the learning problem. Other relevant directions of work include theoretical evaluation of our relaxation or the definition of the set $\mathcal{J}$ (as discussed in Appendix D, $\mathcal{J}$ may contain irrelevant or redundant interventional distributions, and an algorithm that selects relevant interventions may take advantage of ideas such as minimal intervention sets (Lee and Bareinboim, 2018)).

Acknowledgments

TD acknowledges support from a UKRI Turing AI acceleration Fellowship [EP/V02678X/1]. The WMG data was undertaken as part of the NEXTRODE project funded by The Faraday Institution, UK [Grant Number: FIRG015]. The authors would also like to thank Dr Michael Lain at WMG for supporting with collecting battery coating data.

References

Elias Bareinboim, Juan D Correa, Duligur Ibeling, and Thomas Icard. *On Pearl's Hierarchy and the Foundations of Causal Inference*. 2022.

Sander Beckers and Joseph Y Halpern. Abstracting causal models. In *Proceedings of the aaai conference on artificial intelligence*, volume 33, pages 2678–2685, 2019.

Sander Beckers, Frederick Eberhardt, and Joseph Y Halpern. Approximate causal abstractions. In *Uncertainty in Artificial Intelligence*, pages 606–615. PMLR, 2020.

Krzysztof Chalupka, Frederick Eberhardt, and Pietro Perona. Causal feature learning: an overview. *Behaviormetrika*, 44(1):137–164, 2017.

Thomas M Cover. *Elements of information theory*. John Wiley & Sons, 1999.

Ricardo Pinto Cunha, Teo Lombardo, Emiliano N Primo, and Alejandro A Franco. Artificial intelligence investigation of nmc cathode manufacturing parameters interdependencies. *Batteries & Supercaps*, 3(1):60–67, 2020.

Thomas G Dietterich. Ensemble methods in machine learning. In *International workshop on multiple classifier systems*, pages 1–15. Springer, 2000.

Tobias Fritz. A synthetic approach to markov kernels, conditional independence and theorems on sufficient statistics. *Advances in Mathematics*, 370:107239, 2020.

Ronald L Graham, Donald E Knuth, and Oren Patashnik. *Concrete mathematics. A foundation for computer science*. Addison-Wesley, Reading, MA, 1994.

Priyank Jaini, Didrik Nielsen, and Max Welling. Sampling in combinatorial spaces with survae flow augmented mcmc. In *International Conference on Artificial Intelligence and Statistics*, pages 3349–3357. PMLR, 2021.

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

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. *Advances in neural information processing systems*, 30, 2017.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

Sanghack Lee and Elias Bareinboim. Structural causal bandits: where to intervene? *Advances in Neural Information Processing Systems*, 31, 2018.

Kailong Liu, Mona Faraji Niri, Geanina Apachitei, Michael Lain, David Greenwood, and James Marco. Interpretable machine learning for battery capacities prediction and coating parameters analysis. *Control Engineering Practice*, 124:105202, 2022.

Melanie Mitchell. Abstraction and analogy-making in artificial intelligence. *Annals of the New York Academy of Sciences*, 1505(1):79–101, 2021.

Mona Faraji Niri, Kailong Liu, Geanina Apachitei, Luis A.A Román-Ramírez, Michael Lain, Dhammika Widanage, and James Marco. Quantifying key factors for optimised manufacturing of li-ion battery anode and cathode via artificial intelligence. *Energy and AI*, 7:100129, jan 2022. doi: 10.1016/j.egyai.2021.100129.

Jun Otsuka and Hayato Saigo. On the equivalence of causal models: A category-theoretic approach. *arXiv preprint arXiv:2201.06981*, 2022.

Christos H Papadimitriou and Kenneth Steiglitz. *Combinatorial optimization: algorithms and complexity*. Courier Corporation, 1998.

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

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

Eigil F Rischel and Sebastian Weichwald. Compositional abstraction error and a category of causal models. In *Uncertainty in Artificial Intelligence*, pages 1013–1023. PMLR, 2021.

Eigil Fjeldgren Rischel. The category theory of causal models. 2020.

L. A. Román-Ramírez, G. Apachitei, M. Faraji-Niri, M. Lain, D. Widanage, and J. Marco. Effect of coating operating parameters on electrode physical characteristics and final electrochemical performance of lithium-ion batteries. *International Journal of Energy and Environmental Engineering*, mar 2022. doi: 10.1007/s40095-022-00481-w.

Paul K Rubenstein, Sebastian Weichwald, Stephan Bongers, Joris M Mooij, Dominik Janzing, Moritz Grosse-Wentrup, and Bernhard Schölkopf. Causal consistency of structural equation models. *arXiv preprint arXiv:1707.00819*, 2017.

Donald B Rubin. Causal inference using potential outcomes: Design, modeling, decisions. *Journal of the American Statistical Association*, 100(469):322–331, 2005.

Bernhard Schölkopf, Francesco Locatello, Stefan Bauer, Nan Rosemary Ke, Nal Kalchbrenner, Anirudh Goyal, and Yoshua Bengio. Toward causal representation learning. *Proc. IEEE*, 109(5): 612–634, 2021. doi: 10.1109/JPROC.2021.3058954.

Michalis K Titsias and Christopher Yau. The hamming ball sampler. *Journal of the American Statistical Association*, 112(520):1598–1611, 2017.

Simon Vandenhende, Stamatios Georgoulis, Wouter Van Gansbeke, Marc Proesmans, Dengxin Dai, and Luc Van Gool. Multi-task learning for dense prediction tasks: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 2021.

Da Xu, Yuting Ye, and Chuanwei Ruan. Understanding the role of importance weighting for deep learning. *arXiv preprint arXiv:2103.15209*, 2021.

Fabio Massimo Zennaro. Abstraction between structural causal models: A review of definitions and properties. In *UAI 2022 Workshop on Causal Representation Learning*, 2022.

Fabio Massimo Zennaro, Paolo Turrini, and Theodoros Damoulas. Towards computing an optimal abstraction for structural causal models. In *UAI 2022 Workshop on Causal Representation Learning*, 2022.

Appendix A. Notation

$\mathcal{M}, \mathcal{M}'$	SCMs
$\mathcal{X}, \mathcal{X}'$	Set of endogenous variables (for $\mathcal{M}$ and $\mathcal{M}'$)
$\mathcal{U}, \mathcal{U}'$	Set of exogenous variables (for $\mathcal{M}$ and $\mathcal{M}'$)
$\mathcal{F}, \mathcal{F}'$	Set of structural functions (for $\mathcal{M}$ and $\mathcal{M}'$)
$P(\mathcal{U})$,$P'(\mathcal{U}')$	Joint distributions for exogenous variables (in $\mathcal{M}$ and $\mathcal{M}'$)
$X, Y, Z, X_i, ..., X', Y', Z', X'_i, ...$	Endogenous variables (in $\mathcal{M}$ and $\mathcal{M}'$)
$\mathbf{X}, \mathbf{Y} ... \mathbf{X}', \mathbf{Y}' ...$	Sets of endogenous variables (in $\mathcal{M}$ and $\mathcal{M}'$)
$\mathcal{M}[X], \mathcal{M}'[X']$	Domain of endogenous variable (in $\mathcal{M}$ and $\mathcal{M}'$)
$f_1, f_2, ..., f'_1, f'_2, ...$	Structural functions (in $\mathcal{M}$ and $\mathcal{M}'$)
$\mathcal{G}_{\mathcal{M}}$	DAG underlying the SCM $\mathcal{M}$
$pa(X)$	Parents of node X
$\iota := do(X_i = x_i)$	Intervention
$\mathcal{M}_\iota, \mathcal{M}'_{\iota'}$	Post-interventional SCMs
$\boldsymbol{\alpha}$	Abstraction tuple
R	Set of relevant nodes
a	Structural-level surjective map
α	Collection of distributional-level surjective maps
$\alpha_{X'}, \alpha_{Y'}, ...$	Distributional-level surjective maps encoded as binary stochastic matrices
$\mu, \nu, \mu', \nu' ...$	Interventional distributions encoded as stochastic matrices
$\mathcal{J}$	Set of pair sets of endogenous variables or set of interventional distributions
$E_{\boldsymbol{\alpha}}(\mathbf{X}', \mathbf{Y}')$	Abstraction error wrt pair set $(\mathbf{X}', \mathbf{Y}')$ or wrt interventional $P'(\mathbf{Y}'\|do(\mathbf{X}'))$
$e(\boldsymbol{\alpha})$	Overall abstraction error for $\boldsymbol{\alpha}$

Appendix B. Algebraic encoding of a SCM

We illustrate here how a SCM $\mathcal{M}$ defined on a finite set of variables with finite domains may be expressed as a collection of sets and stochastic matrices. A more formal treatment is given by Rischel (2020).

Let $\mathcal{M} = \langle \mathcal{X}, \mathcal{U}, \mathcal{F}, P(\mathcal{U}) \rangle$ be a SCM with underlying DAG $\mathcal{G}_{\mathcal{M}}$.

Each endogenous variable $X_i \in \mathcal{X}$ can be immediately associated with its domain, that is, the discrete set:

$$\mathcal{M}[X_i] = \{0, 1, ..., M\}.$$

When considering a subset of variables $\mathbf{X} = \{X_1, X_2, ..., X_k\} \subseteq \mathcal{X}$, the associated set is the Cartesian product of the domain of each variable, that is, $\mathcal{M}[\mathbf{X}] = \mathcal{M}[X_1] \times \mathcal{M}[X_2] \times ... \times \mathcal{M}[X_k]$.

Thanks to the measurability of the structural functions in $\mathcal{F}$, the probability distribution $P(\mathcal{U})$ over the exogenous variables can be pushforwarded over the endogenous variables; this allows us to define a joint distribution $P(\mathcal{X})$ over the endogenous variables (Peters et al., 2017). The joint distribution can then be factored according to the structure of the DAG, allowing us to compute distributions $P(X_i)$ on root nodes and conditional distributions $P(Y_i|pa(Y_i))$ on non-root nodes. A distribution $P(X_i)$ on a M-dimensional set $\mathcal{M}[X_i] = \{0, 1, ..., M\}$ can be represented as a stochastic $M \times 1$ matrix:

$$\begin{bmatrix} p_1 \\ p_2 \\ ... \\ p_M \end{bmatrix},$$

with the constraint that $\sum_{j=1}^{M} p_j = 1$. A distribution $P(Y_i|pa(Y_i))$, where the set associated with $pa(Y_i)$ is M-dimensional set and the set associate with Y_i is N-dimensional, can be represented as a stochastic $N \times M$ matrix:

$$\begin{bmatrix} p_{1,1} & p_{1,2} & \dots & p_{1,M} \\ p_{2,1} & p_{2,2} & \dots & p_{2,M} \\ \dots & \dots & \dots & \dots \\ p_{N,1} & p_{N,2} & \dots & p_{N,M} \end{bmatrix},$$

with the constraint that, for each $0 \leq k \leq M$, $\sum_{j=1}^{N} p_{j,k} = 1$. This matrix can be seen as encoding in each column k a single conditional distribution $P(Y_i|pa(X_i))$ for each of the M values that the conditioning variable $pa(X_i)$ can assume. Notice that, from the joint distribution, we can also derive in the same way a matrix for any other marginal or conditional distribution not necessarily associated with the factorization defined by the DAG.

It is worth noting that this reduction of a SCM to a collection of sets and stochastic matrices does not violate Pearl's hierarchy (Bareinboim et al., 2022) as it does not claim an equivalence between a SCM and its algebraic reduction; indeed, after algebraic reduction, computing interventions and counterfactuals is not possible anymore.

Finally, an abstraction map $\alpha_{X'}$ is a surjective function from a (set of) low-level variable(s) with cardinality M to a high-level variable with cardinality N. Notice that, for surjectivity to be possible, it must hold $M \geq N$. This matrix can again be represented as a stochastic matrix:

$$\begin{bmatrix} \alpha_{1,1} & \alpha_{1,2} & \dots & \alpha_{1,M} \\ \alpha_{2,1} & \alpha_{2,2} & \dots & \alpha_{2,M} \\ \dots & \dots & \dots & \dots \\ \alpha_{N,1} & \alpha_{N,2} & \dots & \alpha_{N,M} \end{bmatrix},$$

with two constraints: (i) functionality: every column k contains a single one and $N-1$ zeros, thus encoding a degenerate deterministic distribution; (ii) surjectivity: every row j contains at least a one.

Through this encoding, all the quantities in the diagram in Definition 4 can be expressed algebraically.

Appendix C. Definition of Jensen-Shannon distance

We recall here the definition of the discrete Kullback–Leibler divergence the discrete Jensen-Shannon distance (Cover, 1999).

Definition 8 (Kullback–Leibler (KL) divergence) *Let p and q be two probability mass functions on the same domain $\mathcal{X}$, with $q(x) > 0 \; \forall x \in \mathcal{X}$. The KL divergence from p to q is defined as:*

$$D_{KL}(p;q) = -\sum_{x \in \mathcal{X}} p(x) \log \frac{p(x)}{q(x)}. \tag{15}$$

Definition 9 (Jensen-Shannon (JSD) distance) *Let p and q be two probability mass functions on the same domain $\mathcal{X}$, with $p(x) > 0$ and $q(x) > 0 \; \forall x \in \mathcal{X}$. The JS distance between p and q is defined as:*

$$D_{JSD}(p;q) = \sqrt{\frac{1}{2} D_{KL}(p;m) + \frac{1}{2} D_{KL}(q;m)}, \tag{16}$$

where $m = \frac{1}{2}p + \frac{1}{2}q$.

Appendix D. Choice of $\mathcal{J}$

The set $\mathcal{J}$ provides the list of interventional distributions (i.e., diagrams or sub-problems) to be considered in the abstraction learning problem.

According to Definition 6, $\mathcal{J}$ contains all disjoint pair sets $(\mathbf{X}', \mathbf{Y}') \in \mathscr{P}(\mathcal{X}') \times \mathscr{P}(\mathcal{X}')$; such a set $\mathcal{J}$ may be computed by taking the Cartesian product $\mathscr{P}(\mathcal{X}') \times \mathscr{P}(\mathcal{X}')$ of the power sets and by selecting only those elements such that $\mathbf{X}' \cap \mathbf{Y}' = \emptyset$. However, this set $\mathcal{J}$, beyond being very large, might contain many pairs corresponding to irrelevant or redundant interventional distributions.

Irrelevant interventional distributions correspond, for instance, to anti-causal interventions; given a DAG where we have $X' \to Y'$, the set $\mathcal{J}$ will contain the set (X', Y') corresponding to the meaningful interventional distribution $P'(Y'|do(X'))$, but also the set (Y', X') corresponding to the anti-causal interventional distribution $P'(X'|do(Y')))$. Redundant interventional distributions correspond to interventions on independent variables; for instance, given a DAG where we have $X' \to Y' \to Z'$, the set $\mathcal{J}$ will contain the set (Y', Z') corresponding to the minimal interventional distribution $P'(Z'|do(Y'))$, but also the set $((X', Y'), Z')$ corresponding to the redundant interventional distribution $P'(Z'|do(X', Y')))$, where Z' is independent from X' once we intervene on Y'.

Devising a proper algorithm that select a minimal set $\mathcal{J}$ of relevant interventional distributions is left to future work, and discussed in Section 7. In our simulations, we manually select meaningful interventional distributions wrt to the application at hand.

Appendix E. Details about the solution approaches

We provide the pseudocode for our joint approach and for the approaches discussed in Section 5(i) and used in Section 6.

E.1. Joint approach

Algorithm 1 presents the pseudocode of our joint learning approach.

The choice of the set $\mathcal{J}$ at step 4 is discussed in Appendix D. Also, notice that setting up the first term of the loss in the loop at step 5 is the core of the algorithm, where knowledge about the models and the abstraction is exploited.

E.2. Independent approach

Algorithm 2 presents the pseudocode of the independent learning approach.

The independent learning approach follows an algorithm similar to the joint learning algorithm; however, it solves an optimization problem for each sub-problem in $\mathcal{J}$. Notice that the majority voting algorithm in step 4 may be substituted by any other aggregation algorithm.

E.3. Sequential approach

Algorithm 3 presents the pseudocode of the sequential learning approach.

The sequential learning approach follows a similar paradigm as the independent learning approach, but instead of learning a same matrix $W_{X'}$ multiple times, it uses the first learned value; thus, no aggregation is required at the end of learning.

Algorithm 1 Joint Learning Algorithm

Require: base model $\mathcal{M}$, abstracted model $\mathcal{M}'$, set of relevant variables R, mapping a, temperature T, trade-off λ, learning rate η, n_e number of learning epochs

Ensure: locally optimal collection α of abstraction maps $\alpha_{X'}$

Instantiate the parameter set $\mathbf{W} = \{\}$ ▷ Setup the params
for $X' \in \mathcal{X}'$ **do**
  Instantiate a matrix $W_{X'}$ with dimension $|\mathcal{M}'[X']| \times |\mathcal{M}[a^{-1}(X')]|$
  Add matrix $W_{X'}$ to $\mathbf{W}$
  end
Instantiate the loss $\mathcal{L}_1 = \sup\{\}$ ▷ Setup the first term in the loss
Retrieve the set $\mathcal{J}$
for $j = (X', Y') \in \mathcal{J}$ **do**
  Add the term $D_{JSD}(t_{col}(W_{X'})\mu; \nu t_{col}(W_{Y'}))$ to $\mathcal{L}_1$
  end
Instantiate the loss $\mathcal{L}_2 = \sum\{\}$ ▷ Setup the second term in the loss
for $W \in \mathbf{W}$ **do**
  Add $\sum_i (1 - \max_j t_{col}(W)_{ij})$ to $\mathcal{L}_2$
  end
Assemble the loss $\mathcal{L} = \lambda\mathcal{L}_1 + \mathcal{L}_2$ ▷ Setup the loss
for n_e **do**
  Optimize by gradient descent $\mathbf{W} = \mathbf{W} - \eta\frac{\partial\mathcal{L}(\mathbf{W})}{\partial\mathbf{W}}$ ▷ Learn
  end

Algorithm 2 Independent Learning Algorithm

Require: base model $\mathcal{M}$, abstracted model $\mathcal{M}'$, set of relevant variables R, mapping a, temperature T, trade-off λ, learning rate η, n_e number of learning epochs

Ensure: locally optimal collection α of abstraction maps $\alpha_{X'}$

Instantiate the parameter set $\mathbf{W} = \{\}$
Retrieve the set $\mathcal{J}$
for $j = (X', Y') \in \mathcal{J}$ **do**
  Setup $\tilde{\mathbf{W}} = \{\}$ ▷ Setup the params
  Instantiate matrices $W_{X'}$ and $W_{Y'}$ and add them to $\tilde{\mathbf{W}}$
  Instantiate $\mathcal{L}_1 = D_{JSD}(t_{col}(W_{X'})\mu; \nu t_{col}(W_{Y'}))$ ▷ Setup the first term in the loss
  Instantiate $\mathcal{L}_2 = \sum_i (1 - \max_j t_{col}(W_{X'})_{ij}) + \sum_i (1 - \max_j t_{col}(W_{Y'})_{ij})$ ▷ Setup the second term in the loss
  Assemble the loss $\mathcal{L} = \lambda\mathcal{L}_1 + \mathcal{L}_2$ ▷ Setup the loss
  for n_e **do**
    Optimize by gradient descent $\tilde{\mathbf{W}} = \tilde{\mathbf{W}} - \eta\frac{\partial\mathcal{L}(\tilde{\mathbf{W}})}{\partial(\tilde{\mathbf{W}})}$ ▷ Learn
  end
  Add solution $\tilde{\mathbf{W}}$ to $\mathbf{W}$
  end
for *repeated* $W_{X'} \in \mathbf{W}$ **do**
  Aggregate by majority voting ▷ Aggregation
  end

Algorithm 3 Sequential Learning Algorithm

Require: base model $\mathcal{M}$, abstracted model $\mathcal{M}'$, set of relevant variables R, mapping a, temperature T, trade-off λ, learning rate η, n_e number of learning epochs

Ensure: locally optimal collection α of abstraction maps $\alpha_{X'}$

```
Instantiate the parameter set W = {}
Retrieve the set 𝒥
for j = (X', Y') ∈ 𝒥 do
    Setup W̃ = {}                                                    ▷ Setup the params
    if W_X' ∉ W then
        Instantiate matrix W_X' and add it to W̃
        else
            Retrieve W_X' from W
        end
    end
    if W_Y' ∉ W then
        Instantiate matrix W_Y' and add it to W̃
        else
            Retrieve W_Y' from W
        end
    end
    Instantiate ℒ_1 = D_JSD(t_col(W_X')μ; νt_col(W_Y'))              ▷ Setup the first term in the loss
    Instantiate ℒ_2 = Σ_i (1 − max_j t_col(W_X')_ij) + Σ_i (1 − max_j t_col(W_Y')_ij)   ▷ Setup the
second term in the loss
    Assemble the loss ℒ = λℒ_1 + ℒ_2                                ▷ Setup the loss
    for n_e do
        Optimize by gradient descent W̃ = W̃ − η ∂ℒ(W̃)/∂W̃          ▷ Learn
    end
    Add solution W̃ to W
end
```

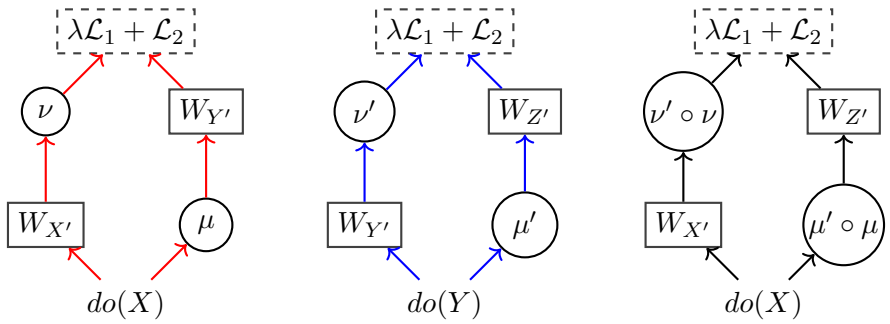


Figure 5: Neural network structure implied by the independent approach. Three separated and independent neural networks are instantiated, each one learning its own parameters. At the end two copies of $W_{X'}, W_{Y'}, W_{Z'}$ will be learned.

Appendix F. Neural network structures

Here we offer a representation of the joint approach and the other approaches as neural networks. Since all the algorithms rely on gradient descent to solve optimization problems connected to the collection of sub-problems specified by $\mathcal{J}$, we can visualize them as neural networks.

In all our depictions, we will depict neural networks for solving an abstraction learning problem with the illustrative set $\mathcal{J} = \{(X', Y'), (X', Z'), (Y', Z')\}$. We will use the following notation: circles represent known interventional distributions encoded in the form of stochastic matrices; solid rectangles represent neural network layers in which we instantiate the learnable parameters W and we process inputs multiplying them by $t_{col}(W)$; finally, the dashed box contains the loss function which produces the learning signal for gradient descent. We use color to highlight the paths belonging to each individual diagram: diagram of (X', Y') in red, diagram of (X', Z') in black, and diagram of (Y', T') in blue.

F.1. Independent approach

Figure 5 shows the structure defined by the independent approach. This approach solves the sub-problems defined by $\mathcal{J}$ independently; therefore, this may be seen as instantiating an independent neural network for each diagram.

F.2. Sequential approach

Figure 6 shows the structure defined by the sequential approach. This approach solves the sub-problems defined by $\mathcal{J}$ independently, but it avoids re-learning weight matrices; this may be seen as instantiating the minimal number of independent neural network to learn all the abstraction maps.

F.3. Joint approach

Figure 2 shows the structure defined by the joint approach. Notice how the individual neural networks defined by the independent or sequential approach get merged into a single neural network with its internal connections and backpropagation defined by $\mathcal{J}$.

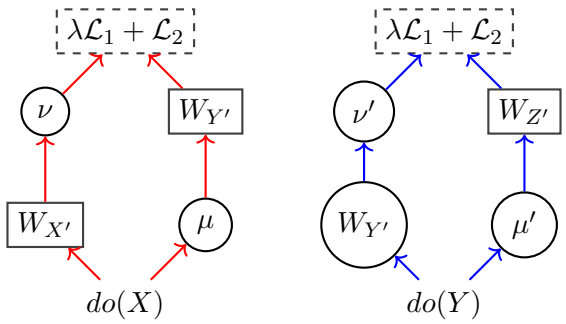


Figure 6: Neural network structure implied by the sequential approach. Only two neural networks are instantiated in order to learn $W_{X'}, W_{Y'}, W_{Z'}$. Notice how, in the second network, the value $W_{Y'}$ is not a learnable parameter anymore (square box) but a fixed matrix (circle).

Appendix G. Details about the synthetic-data simulations

G.1. Synthetic models

Here we provide details about our synthetic models. For each abstraction learning scenario, we define the two SCMs $\mathcal{M}, \mathcal{M}'$ by listing the set of nodes and their domains; we define the abstraction by listing the set R of relevant nodes and the structural-level map a. We illustrate mechanisms and abstractions in a figure, where we follow the convention of representing the base model on top, and the abstracted model on the bottom. We express mechanisms and abstractions as matrices in the figure. A matrix over a solid edge with no source and target X represents distribution $P(X)$. A matrix next to one or more solid edges with sources $X_1, X_2, ..., X_m$ and target Y represents conditional distribution $P(Y|X_1, X_2, ..., X_m)$. A matrix along a dashed edge with target X' represents abstraction $\alpha_{X'}$.

G.1.1. BASIC LUNG CANCER SCENARIO

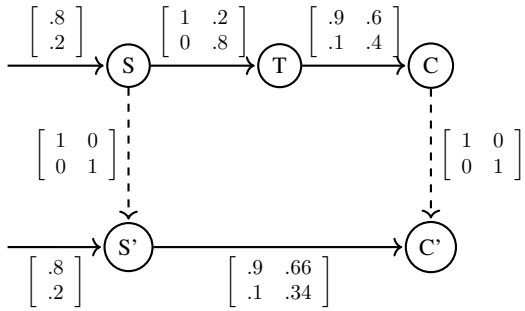
This example is taken from Rischel (2020), and represent a toy lung cancer scenario defined over the variables *smoking* (S, S'), *tar deposits in the lungs* (T), and *lung cancer* (C, C'). The base and the abstracted models are defined on the following nodes:

$$\mathcal{X} = \{S, T, C\} \qquad \mathcal{X}' = \{S', C'\}$$
$$\mathcal{M}[S] = \mathcal{M}[T] = \mathcal{M}[C] = \{0, 1\} \qquad \mathcal{M}'[S'] = \mathcal{M}'[C'] = \{0, 1\}$$

Abstraction is (partially) specified as:

$$R = \{S, C\}$$
$$a(S) = S', a(C) = C'$$

The following figure reports mechanisms and optimal abstractions:



We consider as relevant interventional distributions:

$$P'(C'|do(S')),$$

and, therefore, $\mathcal{J} = \{(S', C')\}$.

With respect to this $\mathcal{J}$, the abstraction has an optimal zero-error $e(\boldsymbol{\alpha}) = 0$ computed by enumeration. Enumeration is feasible since, according to Equation 8, the number of solutions to be evaluated are:

$$|\mathcal{A}| = 2! \left\{ \begin{array}{c} 2 \\ 2 \end{array} \right\} \cdot 2! \left\{ \begin{array}{c} 2 \\ 2 \end{array} \right\} = 4. \tag{17}$$

G.1.2. COLLAPSING LUNG CANCER SCENARIO

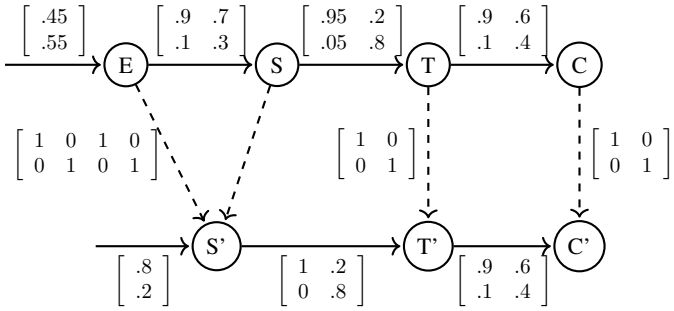
This scenario enriches the base model with a new variable accounting for *environmental factors* (E), and considers an abstraction in which this variable is marginalized away. The base and the abstracted models are defined on the following nodes:

$$\mathcal{X} = \{E, S, T, C\} \qquad \mathcal{X}' = \{S', T', C'\}$$
$$\mathcal{M}[E] = \mathcal{M}[S] = \mathcal{M}[T] = \mathcal{M}[C] = \{0, 1\} \qquad \mathcal{M}'[S'] = \mathcal{M}'[T'] = \mathcal{M}'[C'] = \{0, 1\}$$

Abstraction is (partially) specified as:

$$R = \{E, S, T, C\}$$
$$a(E) = S', a(S) = S', a(T) = T', a(C) = C'$$

The following figure reports mechanisms and optimal abstractions:



We consider as relevant interventional distributions:

$$P'(C'|do(S')),$$
$$P'(C'|do(T')),$$
$$P'(T'|do(S')),$$

and, therefore, $\mathcal{J} = \{(S', C'), (T', C'), (S', T')\}$.

With respect to this $\mathcal{J}$, the abstraction has an optimal non-zero-error $e(\boldsymbol{\alpha}) \approx 0.13$ computed by enumeration. Enumeration is feasible since, according to Equation 8, the number of solutions to be evaluated are:

$$|\mathcal{A}| = 2! \left\{ \begin{array}{c} 4 \\ 2 \end{array} \right\} \cdot 2! \left\{ \begin{array}{c} 2 \\ 2 \end{array} \right\} \cdot 2! \left\{ \begin{array}{c} 2 \\ 2 \end{array} \right\} = 56. \tag{18}$$

G.1.3. EXTENDED LUNG CANCER SCENARIO

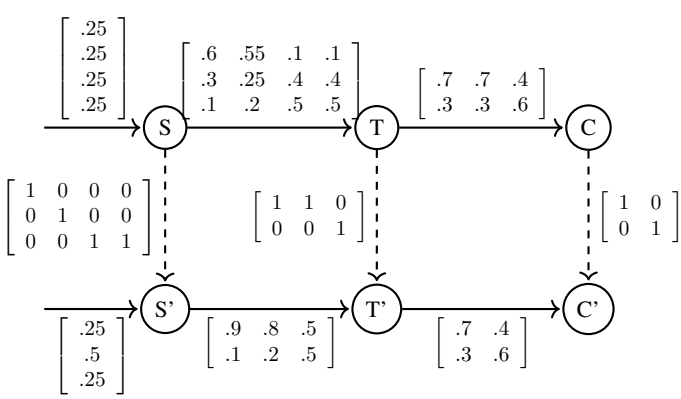
This example constructs a scenario where variables have different cardinalities, and abstraction is focused on reducing the resolution of the experiments. The base and the abstracted models are defined on the following nodes:

$$\mathcal{X} = \{S, T, C\} \qquad \mathcal{X}' = \{S', T', C'\}$$
$$\mathcal{M}[S] = \{0, 1, 2, 3\} \qquad \mathcal{M}'[S'] = \{0, 1, 2\}$$
$$\mathcal{M}[T] = \{0, 1, 2\} \qquad \mathcal{M}'[T'] = \{0, 1\}$$
$$\mathcal{M}[C] = \{0, 1\} \qquad \mathcal{M}'[C'] = \{0, 1\}$$

Abstraction is (partially) specified as:

$$R = \{S, T, C\}$$
$$a(S) = S', a(T) = T', a(C) = C'$$

The following figure reports mechanisms and optimal abstractions:



We consider as relevant interventional distributions:

$$P'(C'|do(S')),$$
$$P'(C'|do(T')),$$
$$P'(T'|do(S')),$$

and, therefore, $\mathcal{J} = \{(S', C'), (T', C'), (S', T')\}$.

With respect to this $\mathcal{J}$, the abstraction has an optimal zero-error $e(\boldsymbol{\alpha}) = 0$ computed by enumeration. Enumeration is feasible since, according to Equation 8, the number of solutions to be evaluated are:

$$|\mathcal{A}| = 3! \left\{ \begin{array}{c} 4 \\ 3 \end{array} \right\} \cdot 2! \left\{ \begin{array}{c} 3 \\ 2 \end{array} \right\} \cdot 2! \left\{ \begin{array}{c} 2 \\ 2 \end{array} \right\} = 432. \tag{19}$$

G.1.4. V-STRUCTURE LUNG CANCER SCENARIO

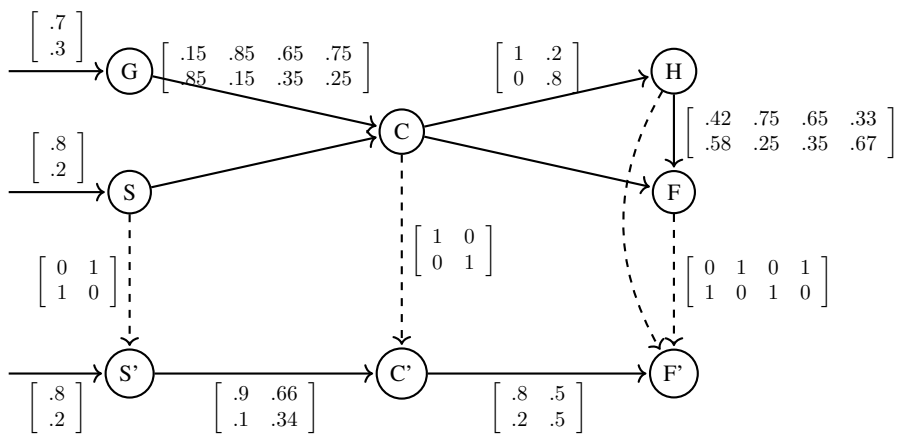
This scenario introduces more complex v-structures, and is partly inspired by the LUCAS toydaset[1]. Beyond the previous variables, new observables are introduced, such as *genetic factors* (G), *coughing* (H), and *fatigue* (F). The base and the abstracted models are defined on the following nodes:

$$\mathcal{X} = \{S, G, C, H, F\} \qquad \mathcal{X}' = \{S', C', F'\}$$
$$\mathcal{M}[S] = \mathcal{M}[G] = \mathcal{M}[C] = \mathcal{M}[J] = \mathcal{M}[F] = \{0, 1\} \qquad \mathcal{M}'[S'] = \mathcal{M}'[C'] = \mathcal{M}'[F'] = \{0, 1\}$$

Abstraction is (partially) specified as:

$$R = \{S, C, H, F\}$$
$$a(S) = S', a(C) = C', a(H) = F', a(F) = F'$$

The following figure reports mechanisms and optimal abstractions:



We consider as relevant interventional distributions:

$$P'(C'|do(S')),$$
$$P'(F'|do(C')),$$
$$P'(F'|do(S')),$$

and, therefore, $\mathcal{J} = \{(S', C'), (C', F'), (S', F')\}$.

With respect to this $\mathcal{J}$, the abstraction has an optimal non-zero-error $e(\boldsymbol{\alpha}) \approx 0.21$ computed by enumeration. Enumeration is feasible since, according to Equation 8, the number of solutions to be evaluated are:

$$|\mathcal{A}| = 2! \left\{ \begin{array}{c} 2 \\ 2 \end{array} \right\} \cdot 2! \left\{ \begin{array}{c} 2 \\ 2 \end{array} \right\} \cdot 2! \left\{ \begin{array}{c} 4 \\ 2 \end{array} \right\} = 56. \tag{20}$$

G.2. Experimental settings

Here we provide details about the experimental settings for all the simulations on synthetic data.

1. http://www.causality.inf.ethz.ch/data/LUCAS.html

G.2.1. COMPARISON WITH BASELINES

In simulation (a) we learn an abstraction on each one of the four synthetic scenarios (*basic, collapsing, extended, v-structure*) using three approaches:

- *Independent approach*: we use the algorithm presented in Appendix E.2. As an aggregation technique (line 4 in Algorithm 2) we use the following algorithm: given multiple solutions $W'_{X'}, W''_{X'}, W'''_{X'}, ..., W^m_{X'}$ for $W_{X'}$, for each column i in $W_{X'}$, we select the index with the highest value among all the candidates, that is, $j^* = argmax_j \left\{ W'_{X'ji}, W''_{X'ji}, W'''_{X'ji} ..., W^m_{X'ji} \right\}$; we then generate a new aggregated solution $\bar{W}_{X'}$ where column i has a one in position j^*, while all the other values in the column are zero.
- *Sequential approach*: we use the algorithm presented in Appendix E.3; we adopt a random ordering of the diagrams in $\mathcal{J}$.
- *Joint approach*: we use the algorithm presented in Appendix E.1.

All algorithms are run with the same settings: temperature $T = 0.1$, trade-off $\lambda = 10$, learning rate $\eta = 0.01$, number of epochs $n_e = 500$. We optimize by gradient descent using the Adam algorithm (Kingma and Ba, 2014). For each approach, we run an ensemble of 10 models, and we select the best wrt the JSD loss term $\mathcal{L}_1$. We repeat experiments 10 times in order to collect reliable statistics.

We measure the performance of the three approaches using different metrics. First of all, we define a set of normalized metrics used to monitor learning at runtime:

- *Normalized JSD loss*: $\frac{1}{|\mathcal{J}_{train}|} \sum_{(\mathbf{X}', \mathbf{Y}') \in \mathcal{J}_{train}} D_{JSD}(\alpha_{\mathbf{Y}'} \mu; \nu \alpha_{\mathbf{X}'})$; this corresponds to the sum of JSDs on each diagram considered during training, divided by the number of such diagrams. Normalization is required because, during training, the independent and joint approach consider all the diagrams $\mathcal{J}_{train} = \mathcal{J}$, but the sequential approach may consider a smaller number of diagrams $\mathcal{J}_{train} \subseteq \mathcal{J}$ (i.e.: the sequential approach may ignore a diagram defined over variables $\alpha_{X'}$ already computed in other diagrams).
- *Normalized surjective penalty*: $\frac{1}{|\mathbf{W}_{train}|} \sum_{W \in \mathbf{W}_{train}} \sum_{i=1}^{M_i} (1 - \max_j t_{col}(W)_{ij})$; this corresponds to the sum of surjective penalties on each weight matrix instantiated during training, divided by the number of such matrices. Normalization is required because, during training, the surjective and joint approach instantiate a number of weight matrices equal to $|\mathbf{W}_{train}| = |\mathcal{X}'|$, but the independent approach may instantiate the same weight matrix multiple times $|\mathbf{W}_{train}| \geq |\mathcal{X}'|$ (i.e.: the independent approach may solve different diagrams in the same $\alpha_{X'}$ independently at the same time).
- *Normalized L1 distance*: $\frac{1}{|\mathbf{W}_{train}|} \sum_{W \in \mathbf{W}_{train}} \ell_1(W - W^*)$, where $\ell_1(W) = \sum_{i,j} |W_{ij}|$ is the ℓ_1-norm, and W^* is the ground-truth optimal solution computed via enumeration; this corresponds to the sum of the ℓ_1 distances from the optimal solution of the current weight matrices, divided by the number of such matrices. As in the case of surjective penalty, normalization is required because, during training, a different number of weight matrices may be instantiated by each approach.

At the end of the training, all the approaches return the collection of learned maps $\mathbf{W}$. We discretize the solutions in $\{0, 1\}$ by rounding, and we evaluate the quality of the computed solution using the following metrics:

- *JSD loss*: $\sum_{(\mathbf{X}',\mathbf{Y}')\in\mathcal{J}} D_{JSD}(\alpha_{\mathbf{Y}'}\mu; \nu\alpha_{\mathbf{X}'})$; this corresponds to the sum of JSDs on all the diagrams in $|\mathcal{J}|$.
- *Surjective penalty*: $\sum_{W\in\mathbf{W}} \sum_{i=1}^{M_i} (1 - \max_j t_{col}(W)_{ij})$; this corresponds to the sum of surjective penalties on all weight matrices in $\mathbf{W}$.
- *L1 distance*: $\sum_{W\in\mathbf{W}} \ell_1(W - W^*)$; this corresponds to the sum of the ℓ_1 distances from the optimal solution of all the weight matrices $\mathbf{W}$.
- *Wallclock time*: we provide an estimate of the running time of each approach. Since our implementation is not parallel, we divide the runtime of the independent approach by the number of diagrams, to simulate the possibility of running each sub-problem in parallel.

G.2.2. EVALUATION OF WEIGHTING

In simulation (b) we learn an abstraction in the *v-structure* scenario. In this scenario, we have defined the set of relevant interventional distribution as $\mathcal{J} = \{(S', C'), (C', F'), (S', F')\}$. We now assume that one intervention (how intervening on smoking affects cancer) is more important than the others (how intervening on smoking affects fatigue, or how intervening on cancer affects fatigue); we also assume that, for the same reason, we have collected a larger number of samples for $P'(C'|do(S'))$. We then impose a weighting schema $\boldsymbol{\kappa} = [2.4, 0.3, 0.3]$ that scales the loss wrt intervention $P'(C'|do(S'))$ more heavily than wrt the other interventions. Notice that the weighting values are chosen to sum up to the same value that the standard uniform weighting $[1, 1, 1]$ would have; keeping the same magnitude is important not to change the ratio between the JSD loss and the surjective penalty.

We run only our algorithm (*joint approach*), with the same setting as in simulation (a).

We measure the quality of the result at the end of training by focusing on *JSD loss*, and comparing the solution to the result obtained without weighting.

G.2.3. ABLATION STUDY

In simulation (c) we learn an abstraction in the *v-structure* scenario without using the surjectivity penalty term $\mathcal{L}_2$ in the loss function.

We run only our algorithm (*joint approach*), with the same setting as in simulation (a).

We measure the quality of the result at the end of training by focusing on *JSD loss* and *L1 distance*, and comparing the solution to the result obtained with the penalty term.

G.2.4. SENSITIVITY ANALYSIS

In simulation (d) we analyze how abstraction learning changes as a function of the hyperparameters T and λ specific to our algorithm.

We run only our algorithm (*joint approach*), with the same setting as in simulation (a), except for the hyperparameter T which is chosen in the set $\{0.01, 0.05, 0.1, 0.5, 1\}$ and the parameter λ which is chosen in the set $\{1, 5, 10, 20, 50\}$.

We measure the quality of the result at runtime by considering *L1 distance* at the end of training.

G.3. Additional results

Figure 7 provides results of simulation (a) on all the remaining scenarios. In terms of ℓ_1-distance from the ground truth, the joint approach achieves performances in line or better than the other approaches.

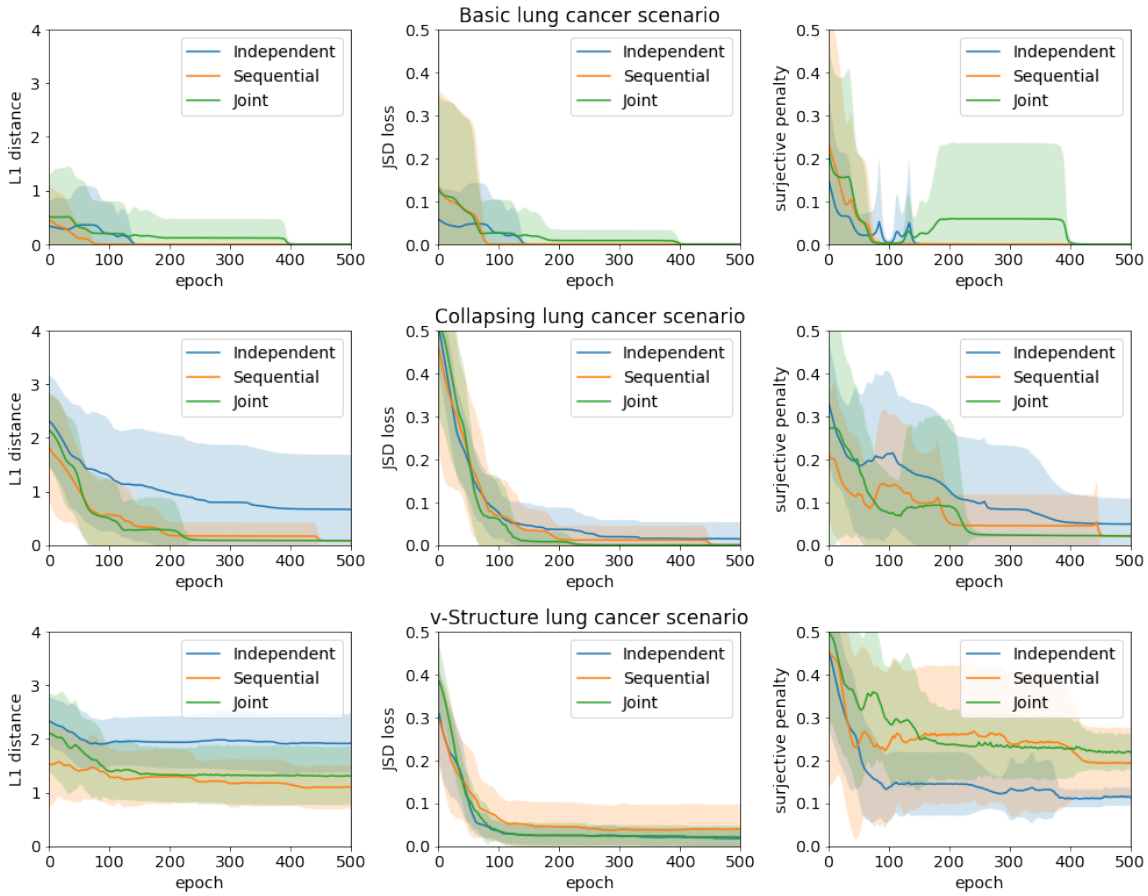


Figure 7: Performance during training on the *basic*, *collapsing*, and *v-structure* scenarios.

Figure 8 shows sample matrices $W_{S'}$ learned with and without surjective penalty. Without surjectivity penalty, $W_{S'}$ simply ignore the value $S' = 0$ in $\mathcal{M}'$ since no value from the base model would be mapped onto it. This could allow the algorithm to achieve a better JSD by reducing the support on which the error is computed. In the limit case, the algorithm may map all the values in the base model onto a single value in $\mathcal{M}'$ in order to reduce JSD. Such a solution would be meaningless, and a surjective penalty prevent this form of collapse.

Figure 9 shows the result of our sensitivity study. In general, the joint algorithm seems to produce reliable results for different combinations of the temperature parameter T and the trade-off parameter λ. We notice, however, that learning is hindered for values of $T \leq 0.05$; this is likely due to an excessively low temperature that prevents a continuous relaxation of the solution space sufficiently smooth to be explored by gradient descent. Low value of trade-off may also have a negative impact on learning; this should likely be ascribed by the surjectivity penalty becoming the leading factor in learning, overshadowing the contribution of JSD.

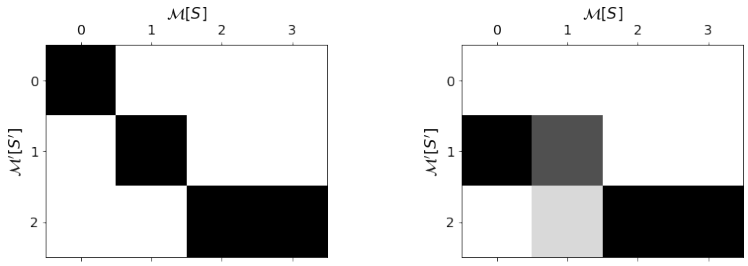


Figure 8: Learned matrix $W_{S'}$ with surjective penalty (left) and without (right).

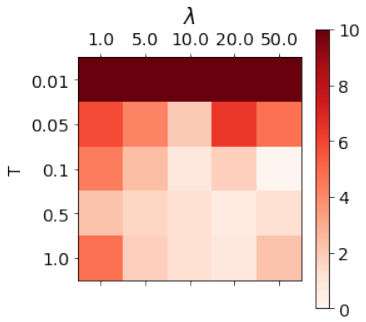


Figure 9: Performance as ℓ_1-distance when using different configuration of T and λ.

Appendix H. Details about the real-world-data simulations

H.1. Data and models

Here we provide details about the real-world data we used, and the models we devised. We use the same graphical conventions used in the Appendix G.1.

H.1.1. LRCS DATASET

The LRCS dataset records the results of a set of experiments investigating the relationship between lithium-ion battery manufacturing parameters for slurry and coating, and target parameters (Cunha et al., 2020). This dataset is publicly available at https://chemistry-europe.onlinelibrary.wiley.com/doi/full/10.1002/batt.201900135.

Each sample in the LRCS dataset represents the result of an experiment, and it is defined by four features:

- *AM Composition:* composition of the slurry as a percentage of active material. This parameter assumes values in $\{92.7, 94, 95, 96\}$;
- *S-to-L ratio:* solid-to-liquid ratio of the slurry as a percentage. This parameter assumes continuous values in $(54, 75)$;
- *Comma gap:* gap in the coating process, measured in $[\mu m]$. This parameter assumes discrete values in $\{50, 75, 100, 200, 300, 400\}$;
- *Viscosity:* viscosity of the slurry, measured in $[Pa \cdot s]$. This parameter assumes continuous values in $(1, 14)$.

Moreover, for each experiment, two target variables are collected:

- *Mass loading:* mass loading of the coating, measured in $\left[\frac{mg}{cm^2}\right]$. This variable assumes continuous values in $(4, 54)$;
- *Porosity:* porosity of the coating as a percentage. This parameter assumes continuous values in $(41, 61)$.

The dataset contains 656 datapoints, that is, 8 datapoints for each one of 82 configurations of control parameters considered.

H.1.2. WMG DATASET

The WMG group has collected a set of recordings from an experiment aimed at measuring the relationship between coating parameters and mass loading. This dataset is available at https://github.com/mattdravucz/jointly-learning-causal-abstraction/.

Each recording contains a large set of parameters defining the state of a coater machine. These parameters are sampled every second in the course of an experiment lasting about three hours. Three values are relevant to us:

- *Comma Bar Operator Position Actual:* basic gap in the coating process, measured in $[mm]$. This parameter, converted to $[\mu m]$, assumes discrete values in $[0, 130]$;
- *Coating Roll Gear Ratio Setpoint:* gap multiplier in the coating process, measured as a percentage. This parameter assumes discrete values in $[100, 150]$;
- *AM Composition:* composition of the slurry as a percentage of active material. This parameter is fixed at 96.

Moreover a target variables is sampled from 800 spatial locations, every eight seconds in the course of the three hours of the experiment:

- *Mass loading:* mass loading of the coating, measured in $\left[\frac{g}{m^2}\right]$. This variable assumes continuous values in $[0, 275)$;

From these recording, we build a dataset performing the following steps:

1. First of all, we convert the recordings (*Comma Bar Operator Position Actual*, *Mass Loading*) to the same unit of measure used in the LRCS dataset.
2. We compute a unique *Comma Gap* measure as a product of *Comma Bar Operator Position Actual* and *Coating Roll Gear Ratio Setpoint*. This transformation is based on specific knowledge about the meaning of the parameters for the WMG coater machine.
3. We subselect the recordings in time; within the three hours of the experiment only short spans of time have a *Comma Gap* actually set to an experimental value; most of the time the *Comma Gap* variable is simply left to zero; we drop all the recordings when the comma gap is zero or when it is just transitioning to an experimental value. At the end, we retain those timesteps when the *Comma Gap* variable is set to an experimental value, together with the corresponding values of *Mass Loading*.
4. We subselect the recordings in space; although mass loading is measured from 800 locations, this information is redundant; instead we consider the 100 central locations (which provide the most reliable measurements) and we average them into $n_{loc} = 2$ spatial measurements.

H.1.3. ALIGNMENT OF LRCS AND WMG DATASET

The two datasets have a clear overlap and strong commonalities in their underlying models. In particular, they share a focus on modelling the casual relation between typical control parameters in the coating process and the resulting mass loading.

We then aim at setting up an abstraction between the underlying models. In order to do this, we will consider the WMG data and its associated model as the low-level model; this is justified by higher spatial resolution of the data (mass loading is measured at multiple locations) and the higher

variable resolution for comma gap. Consequently, we will consider the LRCS data and its associate model as the high-level model.

However, before being able to define SCMs and setup a proper abstraction, we still need to perform further pre-processing to properly align the two datasets. We perform the following operations:

1. From the WMG dataset we drop the control variable *AM Composition*. Since it takes only a single value, it does not bring any information.
2. From the LRCS dataset we drop the control variables *S-to-L ratio* and *Viscosity* which are related to the slurry preparation and not to the coating process.
3. From the LRCS dataset we subselect for *AM Composition*$= 96$ in order to be consistent with the setting of the WMG dataset.
4. We restrict *Comma Gap* values in the LRCS dataset to $\{75, 100, 200\}$, excluding the values $\{50, 300, 400\}$ which are far out of the range considered in the WMG dataset.
5. We extrapolate new values for *Comma Gap* at 75 and 200 in the WMG dataset using a Gaussian process with a linear kernel trained on all the available WMG data. We then define the set of *Comma Gap* values as $\{75, 110, 150, 170, 180, 200\}$
6. For both datasets we discretize the values of *Mass loading* using $n_{bins} = 5$ uniform bins.

At the end of this process, the LRCS dataset contains 64 samples, while the WMG dataset is constituted of 239 samples.

H.1.4. LRCS SCM

The model underlying the LRCS dataset will constitute our abstracted model. We define this SCM making minimal assumptions. The set of endogenous variables in $\mathcal{M}^{LRCS}$ and their associated sets are defined as follows:

$$\mathcal{X}^{LRCS} = \{CG, ML\}$$
$$\mathcal{M}^{LRCS}[CG] = \{75, 100, 200\}$$
$$\mathcal{M}^{LRCS}[ML] = \{0, 1, ..., n_{bins}\}$$

The DAG underlying the model $\mathcal{M}^{LRCS}$ is defined as in Figure 10

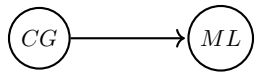


Figure 10: DAG of model $\mathcal{M}^{LRCS}$

Notice that the edge simply expresses the physical causal dependence of *Mass Loading* on the *Comma Gap*. It is also worth noting that this SCM immediately represents the interventional setting in which the control parameter is regulated by an external experimenter.

Critically, as in most real-world scenario, we have no explicit knowledge of the mechanism that determines the variable ML as a function of CG. This mechanism corresponds to a $n_{bins} \times 3$ matrix. We compute this matrix from observed frequencies: for each one of the three possible values of CG we evaluate the empirical distribution of ML into n_{bins}; at the end, we assemble these four empirical distributions into the mechanism matrix.

H.1.5. WMG SCM

The model underlying the WMG dataset will constitute our base model. We follow the same approach used above to define a SCM with minimal assumptions. The set of endogenous variables in $\mathcal{M}^{WMG}$ and their associated sets are defined as follows:

$$\mathcal{X}^{WMG} = \{CG, ML_1, ..., ML_{n_{loc}}\}$$
$$\mathcal{M}^{WMG}[CG] = \{75, 110, 150, 170, 180, 200\}$$
$$\mathcal{M}^{WMG}[ML_i] = \{0, 1, ..., n_{bins}\}$$

The DAG underlying the model $\mathcal{M}^{WMG}$ is defined as in Figure 11.

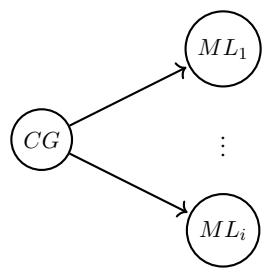


Figure 11: DAG of model $\mathcal{M}^{WMG}$

Again, the edges simply express the physical causal dependence of *Mass Loading* at different locations i on *Comma Gap*.

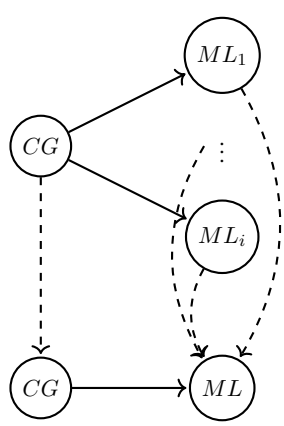
This SCM implies n_{loc} mechanisms, each one encoded into a $n_{bins} \times 6$ matrix. We apply the same approach used before to compute these matrix from the observed frequencies.

H.1.6. ABSTRACTION FROM THE WMG MODEL TO THE LRCS MODEL

Given the two SCMs above, we define a (partial) abstraction $\boldsymbol{\alpha}$ from $\mathcal{M}^{WMG}$ to $\mathcal{M}^{LRCS}$ as follows:

$$R = \{CG, ML_i\}, \forall 1 \leq i \leq n_{loc}$$
$$a(CG) = CG$$
$$a(ML_i) = ML, \forall 1 \leq i \leq n_{loc}$$

This correspond to the following abstraction:



In the abstraction learning problem, we want to learn two weight matrices: the first one corresponds to α_{CG}, and has shape 3×6; the second one corresponds to α_{ML} and has shape $n_{bins} \times (n_{bins})^{n_{loc}}$, in our case 5×25. Notice that, despite the limited number of averaging locations ($n_{loc} = 2$) and discretizing bins ($n_{bin} = 5$), the number of surjective functions to be evaluated in an enumeration algorithm is already unfeasible; indeed, using Equation 8, this value amounts to:

$$|\mathcal{A}| = 4! \left\{ \begin{array}{c} 6 \\ 4 \end{array} \right\} \cdot 5! \left\{ \begin{array}{c} 25 \\ 5 \end{array} \right\} \approx 1.6 \cdot 10^{20}. \tag{21}$$

H.2. Experimental settings

We learn an abstraction between $\mathcal{M}^{WMG}$ and $\mathcal{M}^{LRCS}$ using our joint approach algorithm (see pseudocode in Appendix E.1).

We run our algorithm considering all the possible combinations of hyperparameters in the following sets: temperature $T = \{0.1, 0.2\}$, trade-off $\lambda = \{5.0, 10.0\}$, and learning rate $\eta = \{0.001, 0.002, 0.005\}$. These values are chosen based on the previous experience on the synthetic simulations. For each setting we run an ensemble of 50 models. Each model is trained for a number of epochs $n_e = 10^4$ and optimized by gradient descent using the Adam algorithm Kingma and Ba (2014). At the end, we select the abstraction achieving the best JSD loss $\mathcal{L}_1$.

We evaluate the quality of the result in two ways. First, we discuss qualitatively the solution; although we do not have a ground truth, we can still comment on the pattern of the learned matrix comparing our expectations with the results of learning. Second, we assess quantitatively whether transporting WMG data to the LRCS format, and integrating them with the existing LRCS data, may improve predictions. To do this, we set a three regression tasks with three different setups:

1. *(a) LRCS only:* we consider only the LRCS data. For each value cg of the CG variable, we train a regression model on all the LRCS samples for which $CG \neq cg$. We then test the model on the LRCS samples for which $CG = cg$.
2. *(b) LRCS plus WMG providing support:* we consider the LRCS and the transported $\boldsymbol{\alpha}$(WMG) data. For each value cg of the CG variable, we train a regression model on all the LRCS samples for which $CG \neq cg$ together with all the samples from $\boldsymbol{\alpha}$(WMG). We then test the model on the LRCS samples for which $CG = cg$.
3. *(c) LRCS plus WMG not providing support:* we consider the LRCS and the transported $\boldsymbol{\alpha}$(WMG) data. For each value cg of the CG variable, we train a regression model on all the LRCS and $\boldsymbol{\alpha}$(WMG) samples for which $CG \neq cg$. We then test the model on the LRCS and $\boldsymbol{\alpha}$(WMG) samples for which $CG = cg$.

Task represents a realistic scenario in which a limited amount of data is used to infer a model that allows us to perform interpolation and extrapolation. Task (b) corresponds to a favorable case in which using abstracted data from another research group provides samples of control variables over which we are interpolating or extrapolating. Task (c) represents instead a harder scenario in which the abstracted data provides more information about the domain of interest, but has no samples for the specific values of the control variable for which we want to perform inference.

Notice that although our target variable is ordinal, we avoid using standard ordinal regression models as they can rarely deal with one value of the target variable being absent from the training data. Instead, since our target variable has been generated through uniform binning, we simply rely on a simple linear regression model with lasso penalty.

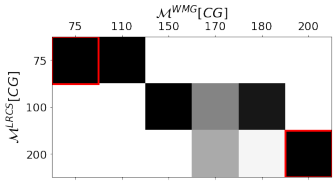


Figure 12: Learned W_{CG}

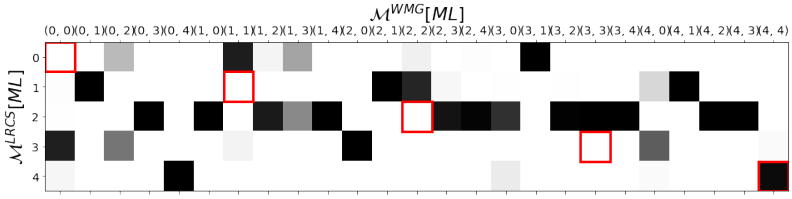


Figure 13: Learned W_{ML}

H.3. Additional results

Figure 12 shows the matrix W_{CG} learned by the joint algorithm. The pattern of this matrix has been described in the main text. It is worth to point out that, formally, other binary matrices could be learned by our algorithm. A permutation of the values in the matrix W_{CG}, accompanied by another permutation in W_{ML}, could still achieve a very low loss value. Our expectation is mainly led by the semantics of the values to be mapped from the base model to the abstracted model: we expect identical values to be mapped to each other. Although this solution may indeed be the global optimum, the algorithm may stop in a local optimum with a vary low abstraction error which counter-intuitively maps values from the base model onto values in the abstracted model.

Figure 13 shows the matrix W_{ML} learned by the joint algorithm. The pattern of this matrix is more difficult to interpret. This follows from a couple of considerations. First, while CG is the variable we intervene upon, ML is the variable we observe conditioned on the intervention. The mapping W_{ML} has then to account for the mechanisms in $\mathcal{M}^{WMG}$ and $\mathcal{M}^{LRCS}$, making an intuitive mapping (highlighted again by the red border) less likely. In other words, the matrix W_{ML} is first of all the matrix that minimizes the JSD and makes the abstraction diagram as commutative as possible; as soon as some noise is introduced in the mechanisms (as it is in our approximate case), the matrix W_{ML} is affected. Second, many values in the domain given by the Cartesian product of measurement at the two spatial locations in the base model are never realized (e.g.: a measurement corresponding to (0,4)); such values end up being unconstrained and they can assume any value as they do not affect the JSD loss of the learning algorithm.