CAUSALSTRUCTCODEC: CAUSALLY-AWARE OBSERVATIONAL AND INTERVENTIONAL DATA GENERATOR

Louis Hernandez
Craft.AI
louis.hernandez@craft.ai

Matthieu Boussard
Craft.AI
matthieu.boussard@craft.ai

ABSTRACT

Over the last few years, causal generative models have massively gained popularity. Their main goal is to generate observational, interventional and counterfactual data. They are also interesting for causal discovery or fair Machine Learning. These generators are based on typical data generation architectures, such as VAEs, GANs or normalizing flows. However, a recent data generation architecture, named Codecs, is more computationally efficient and allows for complex data generation. In this work, we introduce the CausalStructCodecs (CSC), a novel causally-aware architecture based on a specific codec, the StructCodec. We show that results for non-complex data are level with state-of-the-art models for observational and interventional data generation, in significantly fewer epochs.

1 INTRODUCTION

Generative models are useful tools for data augmentation or private data sharing. In the realm of causality, Structural Causal Models (SCM)(Pearl, 2009) are theoretical models representing the true data-generating process. They allow for the computation of interventional and counterfactual generation. But these models are theoretical and, in practice, we do not have access to the functions of the SCM that truly generate the data. We can however have access to the causal graph (Pearl, 2009), using domain knowledge. Dedicated generative models have emerged to leverage that knowledge of the causal graph to generate data, mimicking an SCM, such as MultiCVAE (Karimi et al., 2020), CAREFL (Khemakhem et al., 2021) or VACA (Sanchez-Martin et al., 2021). However, we found that these models have some limitations, including computational efficiency and lack of flexibility regarding complex data structures. Therefore, we provide the CausalStructCodecs (CSC), a novel architecture, based on Codecs(Canale et al., 2022), designed to tackle these issues. We show that it generates interventional and observational data with at least equivalent accuracy to state-of-the-art models while taking advantage of the performance and flexibility of the Codecs architecture.

2 THE ARCHITECTURE

Like other Codecs, CSC can be defined as a quadruplet $C = (E, D, S, L)$, defining an encoder E, a decoder D, a Sampler S and a loss L. These parts can be found on Figure 1, and the details will be explained in the next paragraphs. However, for the sake of conciseness, we invite the reader to refer to Canale et al. (2022) for details about the Codecs, since our architecture is inspired by it. Our main contributions are two-fold, as indicated figure 1, firstly (**1**, in figure 1), we remove the transformer in the encoder, and replace it with a MLP for the embedding, secondly (**2**, in figure 1), we use a mask on the decoder of the transformer to take into account the causal graph. We provide an algorithm for interventional sampling as well.

Pre-treatment Let $(x_1, ..., x_n)$ be an input composed of categorical and numerical features, with its causal graph $\mathcal{G}$ (the same for every data point). Like a StructCodec (Canale et al., 2022), CSC processes features one after the other, and can not process a child without having processed their parent. Thus, the features are topologically sorted according to $\mathcal{G}$ before being used as input to the Encoder. However, unlike CAREFL, which uses only the topological sort and not the full DAG, the full information of the causal graph is retained and used in the transformers.

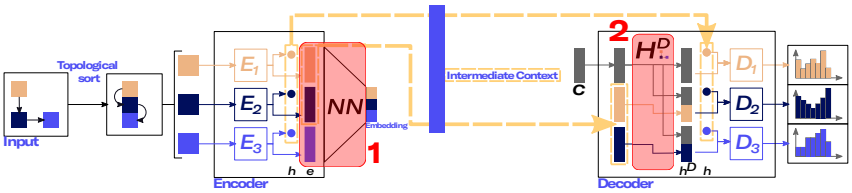

Figure 1: Architecture of the CausalStructCodec

Encoder First, the CSC encoder E calls the encoders of the sub codecs E_k to process each feature k individually and returns for each k, $E_k(x_k) = (e_k, h_k)$ with e_k the encoding and h_k the intermediate context. While the original StructCodecs uses a transformer in its encoder, we do not use it here. Indeed, it would be redundant with the other transformer, and induce computation in contradiction to the causal DAG. To solve this issue, we add a Dense Neural Network (NN Figure 1) to output an embedding, which will not be used here but might be when CSC is used as a sub-structure for larger codecs, like ListCodecs, StructCodecs,(Canale et al., 2022) or even CSC itself. Then, unlike in original StructCodecs, the intermediate context used by the decoder is $\{e_i\}_{i \notin T(\mathcal{G})}$, where $T(\mathcal{G})$ is the set of terminal nodes of $\mathcal{G}$, along with the individual intermediate context h.

Decoder The decoder uses the non-terminal nodes as input for a causal transformer, defined as self attention layers with causal masking, H^D. This transformer takes advantage of the causal graph to ensure that for each k, the decoded distribution of x_k is only computed from the parents and from the conditioning vector c, a vector used as an input for every codec, that only varies when the codec is used as a sub-structure. This step can be written as

$$h_k^D = H^D(c, \{e_i\}_{i \in pa(\mathcal{G},k)}) \tag{1}$$

Where $pa(\mathcal{G}, k)$ is the set of parents of k in $\mathcal{G}$. Finally, this last vector h_k^D is used as a conditioning vector for the final decoders, with the intermediate context h_k, to obtain the final distribution representation $d_k = D_k(h_k^D, h_k)$.

Sampler After being trained following the previous steps, CSC uses its sampler to generate interventional and observational data. The variables are sampled autoregressively. The first feature is sampled from $d_1 = H^D(c)$, then the last sampled feat ure x_k is encoded as (e_k, h_k), which goes through the whole model again to obtain its children in the causal graph. To simulate an intervention $do(x_k)$, instead of sampling x_k from the intervention obtained from its parents, one can simply set it to a value, and let that value run through the model again.

Loss The loss can be computed using d_k and x_k, using $L = -\log(P(x_k|d_k))$.

3 Results

We evaluated the capacity of our model to generate interventional and observational data using synthetic and semi-synthetic datasets from Sanchez-Martin et al. (2021). We found that in all tasks, our model performs similarly to VACA, and better than CAREFL and MultiCVAE, as can be seen in Table 1. Additionally, following Canale et al. (2022), we have reasons to believe our method is more computationally efficient, which is in part confirmed by the low number of epochs necessary to reach the best performance here.

	loan		collider		mgraph		
MMD	Obs	Int	Obs	Int	Obs	Int	Number of epochs
MultiCVAE	5.97 ± 0.21	11.25 ± 0.19	1.04 ± 0.21	14.67 ± 0.19	1.32 ± 0.12	4.07 ± 0.11	448.3 ± 0.31
CAREFL	2.19 ± 0.12	3.42 ± 0.11	2.77 ± 0.22	1.99 ± 0.20	5.33 ± 0.32	4.96 ± 0.29	291.93 ± 49.91
VACA	$\mathbf{0.64 \pm 0.05}$	$\mathbf{1.07 \pm 0.04}$	0.41 ± 0.06	$\mathbf{0.29 \pm 0.06}$	0.20 ± 0.04	$\mathbf{0.69 \pm 0.04}$	373.9 ± 22.56
CSC	1.18 ± 0.05	2.34 ± 0.04	$\mathbf{0.31 \pm 0.06}$	0.45 ± 0.06	$\mathbf{0.17 \pm 0.03}$	0.70 ± 0.03	$\mathbf{7.67 \pm 1.35}$

Table 1: Comparison of CSC with other causally-aware data generators

4 CONCLUSION

We introduced a novel architecture for causal data generation and performances that are equivalent to or better than state-of-the-art models. Future work could include causal discovery, like CAREFL in Khemakhem et al. (2021). Furthermore, alternative transformers could potentially improve the results presented here. Finally, this model is still missing a method to generate counterfactual data.

URM STATEMENT

The authors acknowledge that the first author of this work meets the URM criteria of the ICLR 2023 Tiny Papers Track. As a young (25 years old) master's student, that does not have and is not pursuing a PhD, we believe the first author of this work meets the URM criteria. Furthermore, this work has been done in a small research team (3 people, only one with a PhD) in a start-up, with the first author of this work being the only one focused on causality.

REFERENCES

Luca Canale, Nicolas Grislain, Grégoire Lothe, and Johan Leduc. Generative Modeling of Complex Data, February 2022. URL http://arxiv.org/abs/2202.02145. arXiv:2202.02145 [cs, stat].

Dheeru Dua and Casey Graff. UCI machine learning repository, 2017. URL https://archive.ics.uci.edu/ml/datasets/statlog+(german+credit+data).

Amir-Hossein Karimi, Julius von Kügelgen, Bernhard Schölkopf, and Isabel Valera. Algorithmic recourse under imperfect causal knowledge: a probabilistic approach. In *Advances in Neural Information Processing Systems*, volume 33, pp. 265–277. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/hash/02a3c7fb3f489288ae6942498498db20-Abstract.html.

Ilyes Khemakhem, Ricardo Pio Monti, Robert Leech, and Aapo Hyvärinen. Causal Autoregressive Flows, February 2021. URL http://arxiv.org/abs/2011.02268. arXiv:2011.02268 [cs, stat].

Judea Pearl. *Causality: Models, Reasoning and Inference*. Cambridge University Press, USA, 2nd edition, 2009. ISBN 0-521-89560-X.

Pablo Sanchez-Martin, Miriam Rateike, and Isabel Valera. VACA: Design of Variational Graph Autoencoders for Interventional and Counterfactual Queries, October 2021. URL http://arxiv.org/abs/2110.14690. arXiv:2110.14690 [cs, stat].

A EXPERIMENTAL PROTOCOL

The results in Table 1 are obtained following an experimental protocol following Sanchez-Martin et al. (2021). We compare our method with the existing state-of-the-art methods VACA, CAREFL and MultiCVAE. The datasets used here are the non-additive M-Graph, the linear collider, and the loan dataset, inspired by the German dataset (Dua & Graff, 2017). In practice, we use the official implementation of VACA to run the experiments on the concurrent models and run our own experiments on the same training, validation and testing datasets. CSC was implemented in JAX, and the implementation is available on https://github.com/craft-ai/causal-struct-codec after the double-blind review process. For each intervention, we follow the interventions implemented for the VACA paper. We run each experiment, both interventional and observational for ten different seeds. Therefore, the results in Table 1 are the average of these ten runs for observational generation, and for the interventional distribution, it is the average on the ten runs for each of the interventions, so a total of ten times the number of interventions for each dataset : 8 for collider, 14 for loan and 8 for M-graph.