Journal of Machine Learning Research 21 (2020) 1-5 Submitted 3/19; Revised 11/19; Published 03/20

Causal Discovery Toolbox: Uncovering causal relationships in Python

Diviyan Kalainathan* DIVIYAN@FENTECH.AI
FenTech, TAU, LRI, INRIA, Université Paris-Sud
20 Rue Raymond Aron, 75013 Paris, France

Olivier Goudet OLIVIER.GOUDET@UNIV-ANGERS.FR
LERIA, Université d'Angers,
2 boulevard Lavoisier, 49045 Angers, France

Ritik Dutta DUTTA.RITIK@IITGN.AC.IN
IIT Gandhinagar, Gandhinagar, Gujarat 382355, India

Editor: Andreas Mueller

Abstract

This paper presents a new open source Python framework for causal discovery from observational data and domain background knowledge, aimed at causal graph and causal mechanism modeling. The Cdt package implements an end-to-end approach, recovering the direct dependencies (the skeleton of the causal graph) and the causal relationships between variables. It includes algorithms from the 'Bnlearn' (Scutari, 2018) and 'Pcalg' (Kalisch et al., 2018) packages, together with algorithms for pairwise causal discovery such as ANM (Hoyer et al., 2009). Cdt is available under the MIT License at `https://github.com/FenTechSolutions/CausalDiscoveryToolbox`.

Keywords: Causal Discovery, Graph recovery, open source, constraint-based methods, score-based methods, pairwise causality, Markov blanket

1. Introduction

Causal modeling is key to understand physical or artificial phenomena and to guide interventions. Most softwares for causal discovery have been developed in the R programming language (Kalisch et al., 2018; Scutari, 2018), and a few causal discovery algorithms are available in Python e.g. RCC (Lopez-Paz et al., 2015), CGNN (Goudet et al., 2018) and SAM (Kalainathan et al., 2019), while Python supports many current machine learning frameworks such as PyTorch (Paszke et al., 2017).

The **Causal Discovery Toolbox** (Cdt) is an open-source Python package concerned with observational causal discovery, aimed at learning both the causal graph and the associated causal mechanisms from samples of the joint probability distribution of the data. Cdt includes many state-of-the-art causal modeling algorithms (some of which are imported from R), that supports GPU hardware acceleration and automatic hardware detection. A main goal of Cdt is to provide the users with guidance towards end-to-end experiments,

. * This work was done during Diviyan Kalainathan's PhD Thesis at Univ. Paris-Saclay

by including scoring metrics, and standard benchmark data sets such as the "Sachs" data set (Sachs et al., 2005).

Compared to other causal discovery packages, Cdt unifies pairwise and score-based multi-variate approaches within a single package, implementing an step-by-step pipeline approach (Fig. 1).

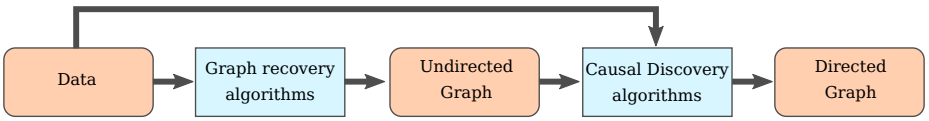

Figure 1: The Cdt causal modeling package: General pipeline

Cdt also provides an intuitive approach for including R-based algorithms, facilitating the task of extending the toolkit with additional R packages. The package revolves around the usage of *networkx.Graph* classes, mainly for recovering (un)directed graphs from observational data. Cdt currently includes 17 algorithms for graph skeleton identification: 7 methods based on independence tests, and 10 methods aimed at directly recovering the skeleton graph. It further includes 20 algorithms aimed at causal directed graph prediction, including 11 graphical and 9 pairwise approaches.

2. Original contributions of the package

The causal pairwise setting considers a pair of variables and aims to determine the causal relationship between both variables. This setting implicitly assumes that both variables are already conditioned on other covariates, or readjusted with a propensity score (Rosenbaum and Rubin, 1983), and that the remaining latent covariates have little or no influence and can be considered as "noise". The pairwise setting is also relevant to complete a partially directed graph resulting from other causal discovery methods. In the 2010s, the pairwise setting was investigated by Hoyer et al. (2009) among others, who proposed the Additive Noise Model (ANM). Later on, Guyon (2013) on Cause-Effect pair (CEP) problems; CEP formulates bivariate causal identification as a supervised machine learning task, where a classifier is trained from examples (A_i, B_i, ℓ_i), where the variable pair (A_i, B_i) is represented by samples of their joint distribution and label ℓ_i indicates the type of causal relationship between both variables (independent, $A_i \rightarrow B_i$, $B_i \rightarrow A_i$). Cdt is one the few packages to include causal pairwise discovery algorithms. These algorithms, mostly implemented using Python or Matlab are often left unmaintained. Therefore, many algorithms that are known to be quite efficient (such as Jarfo (Fonollosa, 2019), first and first in the cause-effect pairs challenges, coded in Python 2.7) are outdated and require a substantial amount of work to fix and update. Cdt implements 9 pairwise algorithms, all coded in Python, 5 of them being new implementations (NCC, GNN, CDS, RECI and a baseline method based on regression error).

The graph setting, extensively studied in the literature, is supported by many packages. Bayesian approaches rely either on conditional independence tests named **constraint-based methods**, such as PC or FCI (Spirtes et al., 2000; Strobl et al., 2017), or on **score-based methods**, involving finding the graph that maximizes a likelihood score through

graph search heuristics, like GES (Chickering, 2002) or CAM (Bühlmann et al., 2014). Other approaches leverage the Generative Network setting, such as CGNN or SAM (Goudet et al., 2018; Kalainathan et al., 2019). Graph setting methods output either a directed acyclic graph or a partially directed acyclic graph. Most approaches in the graph setting are imported from R packages, with the exception of CGNN and SAM.

3. Comparison with other packages

To our best knowledge, **Causality** and **Py-Causal** are the only alternatives to Cdt for causal discovery in Python. However, the only overlap with Cdt concerns the PC-algorithm, common to Py-Causal and Cdt. Akin to Cdt, Py-Causal is a wrapper package but around the Tetrad Java package. Fig. 2 compares the runtimes of the two PC implementations on synthetic graphs with of varying size, connectivity, and number of data points, showing a constant gap in with respect to the number of data points and connectivity of the graph. This gap is due to the creation of the subprocess and the data transfer, that are not taken into account in the PyCausal execution runtime. The gap with respect to the number of nodes is due to different implementations and computational complexity. Further effort will be devoted to imposing the efficiency of our Python-Numba implementation of PC.

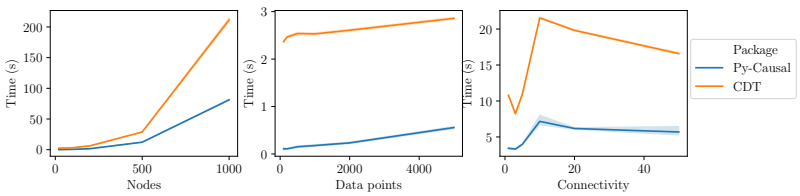

Figure 2: Runtimes of implementations of PC on various graphs

4. Implementation and utilities

R integration. As said, the Cdt package integrate 10 algorithms coded in R and 17 coded in Python. The Cdt package integrates all of them, using Wrapper functions in Python to enable the user to launch any R script and to control its arguments; the R scripts are executed in a temporary folder with a *subprocess* to avoid the limitations of the Python GIL. The results are retrieved through output files back into the main Python process. The whole procedure is modular and allows contributors to easily add new R functions to the package.

Sustainability and deployment. In order for the package to be easily extended, fostering the integration of further community contributions, special care is given to the quality of tests. Specifically, a Continuous Integration tool added to the git repository, allows to sequentially execute tests on new commits and pull request: i) Test all functionalities of the new version on the package on toy data sets; ii) Build docker images and push them to `hub.docker.com` ; iii) Push the new version on *pypi*; iv) Update the documentation

website. This procedure also allows to test the proper functioning of the package with its dependencies.

5. Conclusion and future developments

The Causal Discovery Toolbox (Cdt) package allows Python users to apply many causal discovery or graph modeling algorithms on observational data. It is already used in research projects, such as (Yale et al., 2018; Kalainathan et al., 2019). As the output graphs are `networkx.Graph` classes, these are easily exportable into various formats for visualization softwares, using e.g. Graphviz or Gephi. At the package import, tests are realized to pinpoint the configuration of the user: availability of GPUs and R packages and number of CPUs on the host machine.

The package promotes an end-to-end, step-by-step approach: the undirected graph (bivariate dependencies) is first identified, before applying causal discovery algorithms; the latter are constrained from the undirected graph, with significant computational gains.

Future extensions of the package include: i) reimplementing the R algorithms in Python - Numba and reimplement the Pytorch algorithms in Chainer to drop all heavy dependencies and to integrate Cdt in the Python community with a Numpy-API ; ii) developing GPU-compliant implementation of new algorithms; iii) handling interventional data and time-series data (e.g. for neuroimaging and weather forecast). In the longer term, our priority is to provide the user with tests to whether the standard assumptions (e.g. causal sufficiency assumption) hold and assess the risk of applying methods out of their intended scope.

References

Peter Bühlmann, Jonas Peters, Jan Ernest, et al. CAM: Causal additive models, high-dimensional order search and penalized regression. *The Annals of Statistics*, 2014.

David Maxwell Chickering. Optimal structure identification with greedy search. *Journal of machine learning research*, 3(Nov):507–554, 2002.

José A.R. Fonollosa. Conditional distribution variability measures for causality detection. *Cause Effect Pairs in Machine Learning*, 2019.

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

Isabelle Guyon. Chalearn cause effect pairs challenge, 2013. URL `http://www.causality.inf.ethz.ch/cause-effect.php`.

Patrik O. Hoyer, Dominik Janzing, Joris M. Mooij, Jonas Peters, and Bernhard Schölkopf. Nonlinear causal discovery with additive noise models. In *Neural Information Processing Systems (NIPS)*, pages 689–696, 2009.

Diviyan Kalainathan, Olivier Goudet, Isabelle Guyon, David Lopez-Paz, and Michèle Sebag. Structural agnostic modeling: Adversarial learning of causal graphs. *ArXiv*, 2019.

Markus Kalisch, Alain Hauser, et al. Package 'pcalg'. 2018. URL `https://cran.r-project.org/web/packages/pcalg/index.html`.

David Lopez-Paz, Krikamol Muandet, Bernhard Schölkopf, and Ilya O Tolstikhin. Towards a learning theory of cause-effect inference. In *ICML*, pages 1452–1461, 2015.

Adam Paszke, Sam Gross, Soumith Chintala, et al. Automatic differentiation in PyTorch. 2017. URL `https://pytorch.org/`.

Paul R Rosenbaum and Donald B Rubin. The central role of the propensity score in observational studies for causal effects. *Biometrika*, 70(1):41–55, 1983.

Karen Sachs, Omar Perez, Dana Pe'er, Douglas A Lauffenburger, and Garry P Nolan. Causal protein-signaling networks derived from multiparameter single-cell data. *Science*, 308(5721):523–529, 2005.

Marco Scutari. Package 'bnlearn', 2018. URL `http://www.bnlearn.com/`.

Peter Spirtes, Clark N Glymour, and Richard Scheines. *Causation, prediction, and search*. MIT press, 2000.

Eric V Strobl, Kun Zhang, and Shyam Visweswaran. Approximate kernel-based conditional independence tests for fast non-parametric causal discovery. 2017.

Andrew Yale, Saloni Dash, Ritik Dutta, Isabelle Guyon, Adrien Pavao, and Kristin Bennett. Privacy preserving synthetic health data. *ESANN*, 2018.