# MeDIL: A Python Package for Causal Modelling

**Alex Markham** ALEX.MARKHAM@CAUSAL.DEV
*Research Group Neuroinformatics, Faculty of Computer Science, University of Vienna*

**Aditya Chivukula**
*Department of Statistics, Ludwig Maximilian University of Munich*

**Moritz Grosse-Wentrup**
*Research Group Neuroinformatics, Faculty of Computer Science, University of Vienna*
*Research Platform Data Science @ Uni Vienna*
*Vienna Cognitive Science Hub*

## Abstract

We present the `MeDIL` Python package for causal modelling. Its current features focus on (i) nonlinear unconditional pairwise independence testing, (ii) constraint-based causal structure learning, and (iii) learning the corresponding functional causal models (FCMs), all for the class of measurement dependence inducing latent (MeDIL) causal models. MeDIL causal models and therefore the `MeDIL` software package are especially suited for analyzing data from fields such as psychometric, epidemiology, etc. that rely on questionnaire or survey data.

**Keywords:** causal modelling; Python; structure learning; latent variable model; nonlinear independence; edge clique cover; generative adversarial network.

## 1. Introduction

Markham and Grosse-Wentrup (2020) introduce *measurement dependence inducing latent* (MeDIL) causal models. These models have disjoint sets of (unobserved) latent variables and (observed) measurement variables. In order for a set of random variables to be considered measurement variables, it must satisfy the assumption of *strong causal insufficiency*, i.e., none of the measurement variables may (even indirectly) cause one another—thus, any probabilistic dependence between them must be mediated by latent causes. The assumption of strong causal insufficiency is especially applicable in settings such as psychometric instrument questionnaires, and MeDIL causal models can, for example, be thought of as a causally interpretable factor analysis.

Graphically, MeDIL causal models (MCMs) are represented as directed acyclic graphs with disjoint sets of vertices representing the latent and measurement variables, where the measurement variables are represented as sink vertices (i.e., have no outgoing edges). These MCMs can be inferred by sampling a set of measurement variables as follows:

1. perform (nonlinear) independence tests on samples to generate undirected dependency graph (UDG) over measurement variables
2. perform causal structure learning by applying an edge clique cover finding algorithm to the UDG, resulting in a graphical MCM
3. use generative adversarial networks to learn a functional MCM (i.e., learn the functional relations corresponding to edges in the to the graphical MCM)

See (Markham and Grosse-Wentrup, 2020) for more details, supporting theory, and related work for steps 1 and 2, and see (Chivukula et al., 2020) for those of step 3.

## 2. Features

`MeDIL` is a free/libre software package written in Python (Van Rossum and Drake, 2009) and makes extensive use of NumPy (Oliphant, 2006), which is required for all three submodules. For installation instructions, documentation, and examples, visit `https://medil.causal.dev`

We begin with all the necessary `import` statements and generating the sample data set:

```
# for making sample data
import numpy as np
from medil.examples import triangle_MCM
from medil.functional_MCM import gaussian_mixture_sampler
from medil.functional_MCM import MeDILCausalModel  # also used in
↪  step 3

# for step 1
from medil.independence_testing import hypothesis_test,
↪  dependencies, distance

# for step 2
from medil.ecc_algorithms import find_clique_min_cover as find_cm

# for step 3
from pytorch_lightning import Trainer
from medil.functional_MCM import uniform_sampler, GAN

# for visualization
import medil.visualize as vis
from medil.independence_testing import distance


# make sample data
num_latent, num_observed = triangle_MCM.shape

decoder = MeDILCausalModel(biadj_mat=triangle_MCM)
sampler = gaussian_mixture_sampler(num_latent)

input_sample, output_sample = decoder.sample(sampler,
↪  num_samples=10000)
np.save("measurement_data", output_sample)
```

### 2.1 Independence Testing

The `independence_testing` submodule performs permutation-based hypothesis testing using nonlinear distance correlation from the dcor package Carreño (2020).

```
# step 1: estimate UDG
p_vals, null_corr = hypothesis_test(output_sample.T,
↪  num_resamples=100)
```

```
dep_graph = dependencies(null_corr, 0.1, p_vals, 0.1)
# dep_graph is adjacency matrix of the estimated UDG
```

However, any other preferred way of acquiring the (unconditional) pairwise dependencies can be used, and the resulting UDG can be plugged directly into step 2.

### 2.2 Causal Structure Learning

The `ecc_algorithms` submodule provides an implementation of an algorithm for finding a clique-minimal edge clique cover of a given UDG. The result is an biadjacency matrix that provides the minimal number of latent variables and their connections to measurement variables. It only uses NumPy, though part of the implementation contains code adapted from NetworkX (Hagberg et al., 2008) for finding maximal cliques.

```
# step 2: learn graphical MCM
learned_biadj_mat = find_cm(dep_graph)
```

### 2.3 FCM Learning

Given a set of measurement samples and the causal structure learned in step 2, the `functional_MCM` submodule uses generative adversarial networks (GANs) with the maximum mean discrepancy (MMD) loss to learn a nonlinear functional causal model. It defaults to using a uniform distribution for latent variables and a normal distribution for the exogenous variables, but any prior can be specified. The GANs are built using PyTorch Lightning (Falcon, 2019).

```
# step 3: learn functional MCM
num_latent, num_observed = learned_biadj_mat.shape

decoder = MeDILCausalModel(biadj_mat=learned_biadj_mat)
sampler = uniform_sampler(num_latent)

minMCM = GAN("measurement_data.npy", decoder,
 ↪  latent_sampler=sampler, batch_size=100)
trainer = Trainer(min_epochs=1000)
trainer.fit(minMCM)
```

### 2.4 Visualizing and Evaluating Results

The `visualize` submodule uses Matplotlib (Hunter, 2007).

```
# confirm given and learned causal structures match
vis.show_dag(triangle_MCM)
vis.show_dag(learned_biadj_mat)

# compare plots of distance correlation values for given and learned
 ↪  MCMs
```

```
generated_sample = decoder.sample(sampler, 1000)[1].detach().numpy()
generated_dcor_mat = distance(generated_sample.T)

vis.show_obs_dcor_mat(null_corr, print_val=True)
vis.show_obs_dcor_mat(generated_dcor_mat, print_val=True)

# get params for learned functional MCM; replace '0' with any 'i' in {0, ..., 5} to get params for any corresponding M_i
print(decoder.observed["0"].causal_function)
```

## 3. Future Development

Immediate further development will consist of (1) integrating other measures of independence, such as the Hilbert-Schmidt Independence Criterion, and (2) implementing/integrating other exact and heuristic edge clique cover finding algorithms, e.g., for minimizing the number of functions in the MCM instead of the number of latents, or other partial or heuristic solutions for use on very large networks. Future development will depend on the direction of our theoretical causality work, but is likely to include clustering samples coming from mixtures of MCMs, and learning causally consistent transformations between micro- and macro-models.

## References

C. R. Carreño. dcor: Distance correlation and related e-statistics in Python, 2020. URL https://dcor.readthedocs.io/.

A. Chivukula, A. Markham, B. Bischl, and M. Grosse-Wentrup. Learning MeDIL causal models using generative neural networks, 2020. URL https://causal.dev/files/chivukula_thesis.pdf. masters thesis.

W. Falcon. Pytorch lightning. *GitHub. See: https://github.com/PyTorchLightning/pytorch-lightning*, 3, 2019.

A. A. Hagberg, D. A. Schult, and P. J. Swart. Exploring network structure, dynamics, and function using NetworkX. In G. Varoquaux, T. Vaught, and J. Millman, editors, *Proceedings of the 7th Python in Science Conference*, pages 11 – 15, Pasadena, CA USA, 2008.

J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in science & engineering*, 9(3): 90–95, 2007.

A. Markham and M. Grosse-Wentrup. Measurement dependence inducing latent causal models. *Conference on Uncertainty in Artificial Intelligence (UAI)*, 2020. ISSN 2640-3498. URL http://www.auai.org/uai2020/proceedings/244_main_paper.pdf.

T. E. Oliphant. *A guide to NumPy*, volume 1. Trelgol Publishing USA, 2006.

G. Van Rossum and F. L. Drake. *Python 3 Reference Manual*. CreateSpace, Scotts Valley, CA, 2009. ISBN 1441412697.