Quantifying Intrinsic Causal Influence

By quantifying intrinsic causal influence, we answer the question:

> **How strong is the causal influence of an upstream node to a target node that is not inherited from the parents of the upstream node?**

Naturally, descendants will have a zero intrinsic causal influence on the target node. This method is based on the paper:

> Dominik Janzing, Patrick Blöbaum, Atalanti A Mastakouri, Philipp M Faller, Lenon Minorics, Kailash Budhathoki. Quantifying intrinsic causal contributions via structure preserving interventions Proceedings of The 27th International Conference on Artificial Intelligence and Statistics, PMLR 238:2188-2196, 2024

Let's consider an example from the paper to understand the type of influence being measured here. Imagine a schedule of three trains, `Train A, Train B` and `Train C`, where the departure time of `Train C` depends on the arrival time of `Train B`, and the departure time of `Train B` depends on the arrival time of `Train A`. Suppose `Train A` typically experiences much longer delays than `Train B` and `Train C`. The question we want to answer is: How strong is the influence of each train on the delay of `Train C`?

While there are various definitions of influence in the literature, we are interested in the *intrinsic causal influence*, which measures the influence of a node that has not been inherited from its parents, that is, the influence of the noise of a node. The reason for this is that, while `Train C` has to wait for `Train B`, `Train B` mostly inherits the delay from `Train A`. Thus, `Train A` should be identified as the node that contributes the most to the delay of `Train C`.

See the Understanding the method section for another example and more details.

Skip to main content

How to use it

To see how the method works, let us generate some data following the example above:

```
>>> import numpy as np, pandas as pd, networkx as nx
>>> from dowhy import gcm
```

```
>>> X = abs(np.random.normal(loc=0, scale=5, size=1000))
>>> Y = X + abs(np.random.normal(loc=0, scale=1, size=1000))
>>> Z = Y + abs(np.random.normal(loc=0, scale=1, size=1000))
>>> data = pd.DataFrame(data=dict(X=X, Y=Y, Z=Z))
```

Note the larger standard deviation of the 'noise' in X.

Next, we will model cause-effect relationships as a structural causal model using DoWhy's GCM framework and fit it to the data. Here, we are using the auto module to automatically assign causal mechanisms:

```
>>> causal_model = gcm.StructuralCausalModel(nx.DiGraph([('X', 'Y'), ('Y', 'Z')]))
>>> gcm.auto.assign_causal_mechanisms(causal_model, data)
>>> gcm.fit(causal_model, data)
```

Finally, we can ask for the intrinsic causal influences of ancestors to a node of interest (e.g., Z).

```
>>> contributions = gcm.intrinsic_causal_influence(causal_model, 'Z')
>>> contributions
    {'X': 8.736841722582117, 'Y': 0.4491606897202768, 'Z': 0.35377942123477574}
```

Note that, although we use a linear relationship here, the method can also handle arbitrary non-linear relationships.

Interpreting the results: We estimated the intrinsic causal influence of ancestors of Z, including itself, to its variance (the default measure). These contributions sum up to the variance of Z. As we see here, we observe that ~92% of the variance of Z comes from X.

Related example notebooks

Skip to main content

- Causal Attributions and Root-Cause Analysis in an Online Shop

Understanding the method

Let's look at a different example to explain the intuition behind the notion of "intrinsic" causal influence further:

> A charity event is organised to collect funds to help an orphanage. At the end of the event, a donation box is passed around to each participant. Since the donation is voluntary, some may not donate for various reasons. For instance, they may not have the cash. In this scenario, a participant that simply passes the donation box to the other participant does not contribute anything to the collective donation after all. Each person's contribution then is simply the amount they donated.

To measure the intrinsic causal influence of a source node to a target node, we need a functional causal model. For instance, we can assume that the causal model of each node follows an additive noise model (ANM), i.e. $X_j := f_j(\mathrm{PA}_j) + N_j$, where PA_j are the parents of node X_j in the causal graph, and N_j is the independent unobserved noise term. To compute the "intrinsic" contribution of ancestors of X_n to some property (e.g. variance or entropy) of the marginal distribution of X_n, we first have to set up our causal graph, and learn the causal model of each node from the dataset.

Consider a causal graph $X \to Y \to Z$ as in the code example above, induced by the following ANMs.

$$
\begin{aligned}
X &:= N_X \\
Y &:= 2X + N_Y \\
Z &:= 3Y + N_Z \ ,
\end{aligned}
$$

where $N_w \sim \mathcal{N}(0, 1)$, for all $w \in \{X, Y, Z\}$, are standard Normal noise variables.

Suppose that we are interested in the contribution of each variable to the *variance* of the target Z, i.e. $\mathrm{Var}[Z]$. If there were no noise variables, everything can be contributed to the root node X as all other variables would then be its deterministic function. The intrinsic contribution of each variable to the target quantity $\mathrm{Var}[Z]$ is then really the contribution of corresponding noise term.

Skip to main content

To compute "intrinsic" contribution, we also require conditional distributions of Z given subsets of noise variables N_T, i.e., $P_{Z|N_T}$, where $T \subseteq \{X, Y, Z\}$. We estimate them using an ANM. To this end, we have to specify the prediction model from a subset of noise variables to the target. Below, we quantify the intrinsic causal influence of X, Y and Z to $\mathrm{Var}[Z]$ using a linear prediction model from noise variables to Z.

```
>>> from dowhy.gcm.uncertainty import estimate_variance
>>> prediction_model_from_noises_to_target = gcm.ml.create_linear_regressor()
>>> node_to_contribution = gcm.intrinsic_causal_influence(causal_model, 'Z',
>>>                                                       prediction_model_from_no
>>>                                                       attribution_func=lambda
```

Here, we explicitly defined the variance in the parameter `attribution_func` as the property we are interested in.

Skip to main content

Note

While using variance as uncertainty estimator gives valuable information about the contribution of nodes to the squared deviations in the target, one might be rather interested in other quantities, such as absolute deviations. This can also be simply computed by replacing the `attribution_func` with a custom function: