Attributing Distributional Changes

When attributing distribution changes, we answer the question:

> **What mechanism in my system changed between two sets of data? Or in other words, which node in my data behaves differently?**

Here we want to identify the node or nodes in the graph where the causal mechanism has changed. For example, if we detect an uptick in latency of our application within a microservice architecture, we aim to identify the node/component whose behavior has altered. has changed. DoWhy implements a method to identify and attribute changes in a distribution to changes in causal mechanisms of upstream nodes following the paper:

> Kailash Budhathoki, Dominik Janzing, Patrick Blöbaum, Hoiyi Ng. Why did the distribution change? Proceedings of The 24th International Conference on Artificial Intelligence and Statistics, PMLR 130:1666-1674, 2021.

Additionally, for explaining changes in the mean of the target variable (or other mean-like summary statistics), DoWhy implements a multiply-robust causal change attribution method, which uses a combination of regression and re-weighting to make the final estimates less sensitive to estimation error. This method was presented in the following paper:

> Victor Quintas-Martinez, Mohammad Taha Bahadori, Eduardo Santiago, Jeff Mu, David Heckerman. Multiply-Robust Causal Change Attribution Proceedings of the 41st International Conference on Machine Learning, PMLR 235:41821–41840, 2024.

How to use it

Skip to main content

To see how to use the method, let's take the microservice example from above and assume we have a system of four services X, Y, Z, W, each of which monitors latencies. Suppose we plan to carry out a new deployment and record the latencies before and after the deployment. We will refer to the latency data gathered prior to the deployment as `data_old` and the data gathered after the deployment as `data_new`:

```
>>> import networkx as nx, numpy as np, pandas as pd
>>> from dowhy import gcm
>>> from scipy.stats import halfnorm
```

```
>>> X = halfnorm.rvs(size=1000, loc=0.5, scale=0.2)
>>> Y = halfnorm.rvs(size=1000, loc=1.0, scale=0.2)
>>> Z = np.maximum(X, Y) + np.random.normal(loc=0, scale=0.5, size=1000)
>>> W = Z + halfnorm.rvs(size=1000, loc=0.1, scale=0.2)
>>> data_old = pd.DataFrame(data=dict(X=X, Y=Y, Z=Z, W=W))
```

```
>>> X = halfnorm.rvs(size=1000, loc=0.5, scale=0.2)
>>> Y = halfnorm.rvs(size=1000, loc=1.0, scale=0.2)
>>> Z = X + Y + np.random.normal(loc=0, scale=0.5, size=1000)
>>> W = Z + halfnorm.rvs(size=1000, loc=0.1, scale=0.2)
>>> data_new = pd.DataFrame(data=dict(X=X, Y=Y, Z=Z, W=W))
```

Here, we change the behaviour of Z, which simulates an accidental conversion of multi-threaded code into sequential one (waiting for X and Y in parallel vs. waiting for them sequentially). This will change the distribution of Z and subsequently W.

Next, we'll model cause-effect relationships as a probabilistic causal model:

```
>>> causal_model = gcm.ProbabilisticCausalModel(nx.DiGraph([('X', 'Z'), ('Y', 'Z')
>>> gcm.auto.assign_causal_mechanisms(causal_model, data_old)
```

Finally, we attribute changes in distributions of W to changes in causal mechanisms:

```
>>> attributions = gcm.distribution_change(causal_model, data_old, data_new, 'W')
>>> attributions
{'W': 0.012553173521649849, 'X': -0.007493424287710609, 'Y': 0.0013256550695736396
```

Although the distribution of W has changed as well, the method attributes the change almost

Skip to main content

The multiply-robust method works similarly:

```
>>> attributions_robust = gcm.distribution_change_robust(causal_model, data_old, d
>>> attributions_robust
{W: 0.012386916935751025, X: -0.0002994129507127999, Y: 0.006618489296759587, Z: 0
```

As before, we estimate a large attribution score for Z, and a score close to 0 for the other variables. Note that the unit of the attribution scores depends on the used measure (see the next section). By default, `distribution_change` decomposes changes in the KL divergence to the original distribution, whereas `distribution_change_robust` decomposes changes in the mean of the target node W.

As the reader may have noticed, there is no fitting step involved when using this method. The reason is, that this function will call `fit` internally. To be precise, this function will make two copies of the causal graph and fit one graph to the first dataset and the second graph to the second dataset.

Related example notebooks

- Finding the Root Cause of Elevated Latencies in a Microservice Architecture
- Causal Attributions and Root-Cause Analysis in an Online Shop
- Finding Root Causes of Changes in a Supply Chain
- ../../../example_notebooks/gcm_cps2015_dist_change_robust.ipynb

Understanding the method

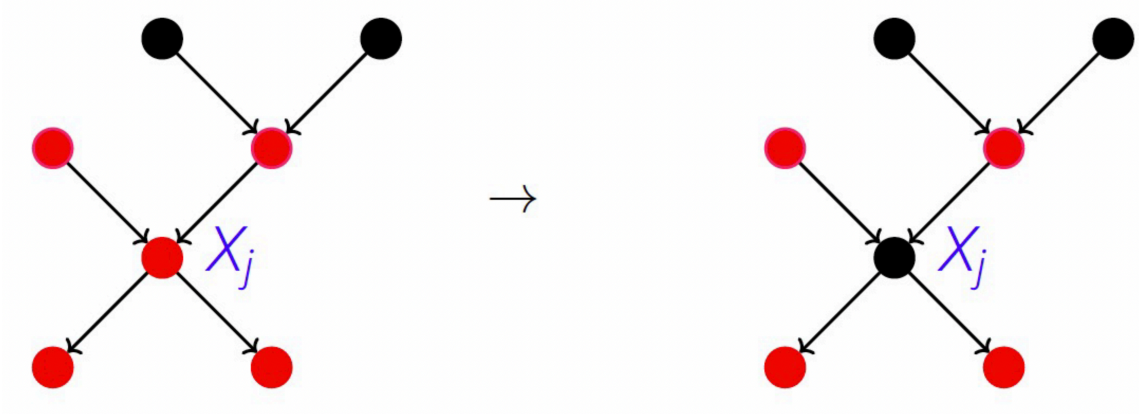
The idea behind these methods is to *systematically* replace the causal mechanism learned based on the old dataset with the mechanism learned based on the new dataset. After each replacement, new samples are generated for the target node, where the data generation process is a mixture of old and new mechanisms. Our goal is to identify the mechanisms that have changed, which would lead to a different marginal distribution of the target, while unchanged mechanisms would result in the same marginal distribution. To achieve this, we employ the idea of a Shapley symmetrization to systematically replace the mechanisms. This enables us to identify which nodes have changed and to estimate an attribution score with respect to some measure. Note that a change in the mechanism could be due to a functional change in the underlying model

Skip to main content

or a change in the (unobserved) noise distribution. However, both changes would lead to a change in the mechanism.

The main difference between `distribution_change` and `distribution_change_robust` is in how the attribution measures are computed. `distribution_change` first learns the conditional distributions of each node in the 'old' and in the 'new' data, and then uses these estimates to compute the attribution measure. `distribution_change_robust` does not learn the entire conditional distributions, but rather the conditional means (regression) and importance weights (re-weighting), which are then combined to compute the attribution measure. The final step, using Shapley values to combine these attribution measures into a meaningful score, is the same for both methods.

The steps in `distribution_change` are as follows:



1. Estimate the conditional distributions from 'old' data (e.g., latencies before deployment): $P_{X_1,\dots,X_n} = \prod_j P_{X_j|PA_j}$, where $P_{X_j|PA_j}$ is the causal mechanism of node X_j and PA_j the parents of node X_j
2. Estimate the conditional distributions from 'new' data (e.g., latencies after deployment): $\tilde{P}_{X_1,\dots,X_n} = \prod_j \tilde{P}_{X_j|PA_j}$
3. Replace mechanisms based on the 'old' data with mechanisms based on the 'new' data systematically, one by one. For this, replace $P_{X_j|PA_j}$ by $\tilde{P}_{X_j|PA_j}$ for each j. If nodes in $T \subseteq \{1,\dots,n\}$ have been replaced before, we get $\tilde{P}_T^{X_n} = \sum_{x_1,\dots,x_{n-1}} \prod_{j\in T} \tilde{P}_{X_j|PA_j} \prod_{j\notin T} P_{X_j|PA_j}$, a new marginal for node n.
4. Attribute the change in the marginal given T to X_j using Shapley values by comparing $P_{T\bigcup\{j\}}^{X_n}$ and $P_T^{X_n}$. Here, we can use different measures to capture the change, such as KL divergence to the original distribution or difference in variances etc.

Skip to main content

For more detailed explanation, see the corresponding paper: Why did the distribution change?

The steps in `distribution_change_robust` are the following:

1. Learn the regression functions: Using a regression algorithm, we estimate the dependence between each node and its parents in the 'new' data if we wish to shift its causal mechanism, or in the 'old' data, if we wish to keep that node unchanged.
2. Learn the importance weights: Using a classification algorithm, we estimate importance weights that place higher weight on data points where a given causal mechanism resembles the 'new' data more.
3. A combination of the regressions and the weights allows us to estimate the mean of the target node under the distribution $\tilde{P}_T^{X_n} = \sum_{x_1,\ldots,x_{n-1}} \prod_{j\in T} \tilde{P}_{X_j|PA_j} \prod_{j\notin T} P_{X_j|PA_j}$, where the causal mechanisms in $T \subseteq \{1,\ldots,n\}$ have been shifted to be as in the 'new' data.
4. Attribute the change in the marginal given T to X_j using Shapley values by comparing $P_{T\bigcup\{j\}}^{X_n}$ and $P_T^{X_n}$.

For more detailed explanation, see the corresponding paper: Multiply-Robust Causal Change Attribution

Previous Next