On testing generalized independence constraints

Generalized independence constraints are dormant independences that appear in interventional distributions that can be obtained from the original observed data distribution. Consider the acyclic directed mixed graph (ADMG) below and to the left, where the bidirected edge indicates unmeasured common causes that have been marginalized out.

You can check that there is only one ordinary conditional independence in this graph – $A \perp\!\!\!\perp C|B$. However, there is also a missing edge between variables A and Y. The question is, does this edge correspond to a constraint on the observed data distribution, and if so, why might it be useful to find such a constraint? Imagine that Y is some outcome that we are interested in. The absence of a direct edge $A \to Y$ implies that all of the effect of A on Y is mediated through the variables B and C. If A is something that is hard or expensive or even unethical to intervene on, confirming the absence of a direct effect, allows us to rest easy in the knowledge that it is sufficient to intervene on something easier, say B or C, for an equivalent impact on the outcome Y.

So how does one go about testing for such a constraint? One way is to use parametric likelihoods and compare the Bayesian Information Criterion of the models with and without the $A \to Y$ edge. We leave further discussion of that to another time. For now, we discuss a non-parametric method of testing the same, that only relies on access to an appropriate conditional independence test. First imagine going to an interventional distribution where we intervene on C, i.e., $p(ABY|\,\mathrm{do}(c))$. This corresponds to the graph right next to the original ADMG where C is now "fixed" or intervened upon. We call such a graph a conditional ADMG (CADMG). Operationally, how do we obtain this interventional distribution? We block all backdoor paths from C to its descendants. This corresponds to dividing by $p(C|B)$, and can be thought of as g-formula or truncated factorization in the sense that this term has been dropped from the original factorization, or as IPW in the sense that each sample can be inverse-weighted using the probability $p(C|B)$. Using either interpretation, we arrive at a distribution that is Markov to the CADMG above, and we can see that there is a dormant independence in this CADMG – $A \perp\!\!\!\perp Y$ in the distribution $p(ABCY)/p(C|B)$. We use the IPW interpretation in the code below to demonstrate how one can test for a generalized independence constraint in practice, and discuss some practical issues after.

Code

```
In [1]: # imports (numpy/scipy/sklearn are standard
        # but the other probably needs to be installed)
        import numpy as np
        from scipy.special import expit
        from gsq.ci_tests import ci_test_bin
        from sklearn.linear_model import LogisticRegression
```

In [2]:
```
np.random.seed(42)
# Generate data from A->B->C->Y, B<->Y
n = 500000
A = np.random.binomial(1, 0.5, n)
U1 = np.random.uniform(0, 1, n)
U2 = np.random.uniform(1, 2, n)
B = np.random.binomial(1, expit(-2*A - 2.5*U1 + 2*U2), n)
C = np.random.binomial(1, expit(-0.5 + 1.2*B), n)
Y = np.random.binomial(1, expit(-0.2 + 1.5*C + 2*U1 - 1.5*U2), n)

# Make a data matrix
data = np.column_stack((A, B, C, Y))

# We first test for the ordinary conditional independence of A _||_ C | B
# Note: ci_test_bin is a chi-square test for conditional independence
# that takes a data matrix, a column index for each of the variables we are
# testing independence for, and a set of column indices for the conditioning variables
print("Cannot reject the null hypothesis that A _||_ C | B: p-value=",
      round(ci_test_bin(data, 0, 2, set([1])), 3))

# We also confirm the absence of the independence A _||_ Y | C
print("Can reject the null hypothesis that A _||_ Y | C: p-value=",
      round(ci_test_bin(data, 0, 3, set([2])), 3))
```

```
Cannot reject the null hypothesis that A _||_ C | B: p-value= 0.509
Can reject the null hypothesis that A _||_ Y | C: p-value= 0.0
```

In [3]:
```
# We now test for the generalized indepdendence constraint A _||_ Y after fixing C

# First fit a regression for p(C|B)
model = LogisticRegression(C=1e20, solver='lbfgs', fit_intercept=False)
X = np.column_stack((np.ones((n, 1)), B))
model.fit(X, C)
probs = model.predict_proba(X)[:,1]
idx_C0, idx_C1 = C==0, C==1

# Reweigh and resample
weights = np.ones((n,))
weights[idx_C0] = 1/(1-probs[idx_C0])
weights[idx_C1] = 1/(probs[idx_C1])
weights = weights/weights.sum()
reweighed_idxs = np.random.choice(range(n), size=n, replace=True, p=weights)
# Test with the reweighed dataset
print("Cannot reject the null hypothesis that A _||_ Y after fixing C: p-value=",
      round(ci_test_bin(data[reweighed_idxs,:], 0, 3, set([])), 3))
```

```
Cannot reject the null hypothesis that A _||_ Y after fixing C: p-value= 0.574
```

Practical issues/closing notes

- One may notice the large sample size used in the example code above. Prior works [2, 1] on the topic indicate that large sample sizes or effect sizes are required to (a) detect generalized independence constraints and (b) distinguish it from an ordinary conditional independence constraint $A \perp\!\!\!\perp Y|C$.
- From personal experience in using the BIC scoring method on Gaussian data to distinguish between graphs with and without the constraint, it seems that fitting parametric likelihoods may partially alleviate the problem.
- I use reweighed bootstrap here in order to mimic a reweighed conditional independence test. Ideally, one would reimplement the test itself to accept weighted inputs.
- One may have to iteratively reweigh, i.e., learn a propensity score for another variable in a reweighed distribution and then further reweigh using the newly learned score to arrive at a distribution where the constraint is present.
- I would use the `statsmodel` API over `sklearn` in python. `sklearn` for whatever reason does not allow turning off regularization completely. I only use `sklearn` here because most people already have it installed.

References

[1] Robin J. Evans. Model selection and local geometry. *arXiv preprint arXiv:1801.08364*, 2018.

[2] Ilya Shpitser, Thomas S. Richardson, James M. Robins, and Robin J. Evans. Parameter and structure learning in nested markov models. *arXiv preprint arXiv:1207.5058*, 2012.