Journal of Machine Learning Research 21 (2020) 1-6 Submitted 9/19; Revised 10/20; Published 11/20

algcomparison: Comparing the Performance of Graphical Structure Learning Algorithms with TETRAD

Joseph D. Ramsey JDRAMSEY@ANDREW.CMU.EDU
Department of Philosophy
Carnegie Mellon University
Pittsburgh, PA USA

Daniel Malinsky DSM2128@CUMC.COLUMBIA.EDU
Department of Biostatistics
Columbia University
New York, NY USA

Kevin V. Bui KVB2@PITT.EDU
Department of Biomedical Informatics
University of Pittsburgh
Pittsburgh, PA USA

Editor: Andreas Mueller

Abstract

In this report we describe a tool for comparing the performance of graphical causal structure learning algorithms implemented in the TETRAD freeware suite of causal analysis methods. Currently the tool is available as package in the TETRAD source code (written in Java). Simulations can be done varying the number of runs, sample sizes, and data modalities. Performance on this simulated data can then be compared for a number of algorithms, with parameters varied and with performance statistics as selected, producing a publishable report. The package presented here may also be used to compare structure learning methods across platforms and programming languages, i.e., to compare algorithms implemented in TETRAD with those implemented in MATLAB, Python, or R.

Keywords: causal discovery, graphical models, structure learning, evaluation

1. Introduction

Often researchers are faced with the problem of choosing an algorithm from among possibly dozens of relevant algorithms for a particular task. This can be time-consuming and error-prone; one must try each algorithm in turn, vary the parameters for that algorithm, run it in simulation on common data sets that hopefully reflect the properties of the real data of interest, and discern which algorithm has the best performance over the range of cases under study. Research papers tend to compare only a small number of algorithms at a time on performance statistics that may not be of interest to the user, using simulation settings not appropriate for the domain. Ideally a user could directly compare a range of algorithms on data of their choosing and on performance statistics of interest to them, so that they could make an informed decision as to which algorithm(s) may be best suited to the user's particular purpose.

We focus on the causal structure learning algorithms in the TETRAD freeware.[1] Within TETRAD, we have created a tool for comparing algorithms, both "basic" algorithms with varying parameter settings and algorithms variously combined. The relevant code is contained in the package **algcomparison** within TETRAD.[2] It is possible to construct studies in which combinations of structure learning algorithms are compared head-to-head on common data, with known true models; winners conveniently float to the top of the list of algorithms when sorted by a utility function that reflects the user's interests. Algorithms that perform poorly for the intended type of data analysis quickly become apparent. This makes it easy to identify the general class of algorithms the user may want to select from for their purposes.

In TETRAD, **algcomparison** has available a wide range of algorithms and the flexbility to add new algorithms easily. Combinations of existing algorithms are often treated in practice as novel algorithms; we allow them to be treated as such. **algcomparison** has some standard styles of simulation readily available and the user is able to add new simulation styles. The tool enables a user to use "default" parameters and to change the default settings of the parameters easily. **algcomparison** has a range of built-in standard performance statistics for quantifying the accuracy of a learned structure, and with some straightforward programming the user may add new performance statistics. Finally, in deference to the user's needs, **algcomparison** enables the user to decide which combination of performance statistics to employ to determine the best algorithm or algorithms. This is because different users with different scientific backgrounds may very well have different views as to what is important in an estimated causal model. We take the view that these differences should be handled using a modular architecture. Algorithms, simulations, parameters, performance statistics, independence tests, and scores can be independently input into a central comparison class to execute experiments as the user wishes.

2. Background

The TETRAD software was introduced in the mid-1980s to aid in constructing, testing, predicting with, and learning causal statistical models based on structural equations or graphical representations such as Directed Acyclic Graphs (DAGs) (Glymour et al., 1987; Spirtes et al., 1990; Scheines et al., 1998; Spirtes et al., 2000). The capabilities and flexibility of TETRAD has increased with years of algorithm development and application in several scientific fields including biology (e.g. Shipley et al., 2006), neuroscience (e.g. Smith et al., 2011; Mills-Finnerty et al., 2014), economics (e.g. Bessler and Lee, 2002; Demiralp and Hoover, 2003), climate science (e.g. Ebert-Uphoff and Deng, 2012), education research (e.g. Rau et al., 2013), and other areas. Though TETRAD is capable of performing a wide range of tasks relevent to causal inference, we will focus only on graphical structure learning here.

TETRAD implements numerous algorithms which search for causal graphical models. The resultant models are intended to have a causal interpretations, the precise details of which depend on the underlying assumptions and the type of output graph produced by the method. At the time of writing, there are dozens of structure learning algorithms and variations; we do not review them all in detail here. Recent overviews of causal graphical

1. `https://github.com/cmu-phil/tetrad`
2. The full package path in the code library is `edu.cmu.tetrad.algcomparison`.

modeling, structure learning algorithms, and the assumptions underlying causal inference from observational data can be found in Spirtes and Zhang (2016), Drton and Maathuis (2017), and Heinze-Deml et al. (2018). Popular alternative software packages for structure learning include **pcalg** (Kalisch et al., 2012) and **bnlearn** (Scutari, 2010) in R and the **Bayes Net Toolbox** in MATLAB (Murphy, 2001). These packages implement some of the same algorithms available in TETRAD (e.g., all packages implement some variants of the PC algorithm), as well as some methods which are not currently available in TETRAD (e.g., **pcalg** implements the GIES procedure for learning from interventional data, **bnlearn** implements several variations on Markov Blanket-based algorithms, and **Bayes Net Toolbox** implements an MCMC algorithm for learning DAGs). TETRAD also implements many algorithms not available on any of these alternative platforms (including novel versions of PC, FCI, and GES; a hybrid variant of FCI called GFCI; a hybrid approach to learning from mixed-type data sets called MGM; and several approaches to learning cyclic graphs such as CCD and FASK). Although cross-evaluation of multiple algorithms is possible using (for example) **pcalg** as a base,[3] the benefit of **algcomparison** is its flexibility and the fact that basically no novel programming is required on behalf of the user; the tool can be executed entirely on the command line with simple XML scripts as we describe below.

3. A modular architecture

The source code is structured around several Java interfaces, specifically interfaces for specifying search algorithms (`Algorithm`), simulations and their parameters (`Simulation`), parametric or nonparametric conditional independence tests (`IndependenceWrapper`), model scores (`ScoreWrapper`), and so on, along with a special class called `Comparison` which contains methods to load in files or data and execute a sequence of simulations, algorithms, and comparisons for given parameter settings. A `Simulation` method generates a random graph (e.g., by adding edges between vertices arranged in some partial order with fixed probability) and then generates samples according to that graph by some scheme (e.g., a linear or nonlinear structural equation model, or a multinomial model for discrete data). An `Algorithm` is any method which takes data and parameter settings as input and returns a graphical representation: these may be constraint-based methods which use independence tests (e.g., PC, FCI and their variants), score-based methods which optimize some model score (e.g., Fast Greedy Equivalence Search and related), pairwise methods based on regression residual asymmetries, some novel user-specified algorithm, or a combination of these. The software also includes some methods for learning undirected graphs with no obvious causal interpretation (e.g., graphical Lasso), since these may be useful as subroutines of a causal learning procedure.

4. Running comparison experiments

There are two ways to run comparisons with **algcomparison**. The first is by executing a short Java class within TETRAD, typically in an integrated development environment (IDE). Several example scripts are bundled with the code and may be modified by the user. The second is by running an XML configuration file on the command line, which does not require

3. `https://github.com/christinaheinze/CompareCausalNetworks`

any knowledge of Java programming.[4] In both cases, the modular components described above are combined according to the user's specifications and passed to the `Comparison` class. An example XML configuration file is as follows:

```
<comparison> <compareBy>
        <search>
                <simulations>
                        <simulation source="generate">
                        <graphtype>RandomForward</graphtype>
                        <modeltype>SemSimulation</modeltype>
                        </simulation>
                </simulations>
                <algorithms>
                        <algorithm name="pc-all">
                        <test>fisher-z-test</test>
                        </algorithm>
                        <algorithm name="fges">
                        <score>sem-bic</score>
                        </algorithm>
                </algorithms>
                <parameters>
                <parameter name="numRuns">10</parameter>
                <parameter name="numMeasures">1000</parameter>
                <parameter name="avgDegree">4</parameter>
                </parameters>
        </search> </compareBy>
        <statistics>
                <statistic>adjacencyPrecision</statistic>
                <statistic>adjacencyRecall</statistic>
                <statistic>SHD</statistic>
        </statistics>
</comparison>
```

This generates data from a linear Gaussian model with 1000 variables and compares the PC algorithm with Fisher Z independence test to the FGES algorithm, which uses the BIC score. It calculates adjacency precision, adjacency recall, and Structural Hamming Distance from the true model (its Markov equivalence class) over 10 runs. Running this configuration on the command line would generate a table of results saved to an output file. There is a wide range of algorithms, settings, tests, and statistics available within TETRAD which we do not have space to enumerate here, though a user may generate a list of available options by executing the class called `RunConfig`.

5. Cross-platform comparisons

It may be desirable to assess algorithms implemented in alternative languages and software platforms. Furthermore, one may wish to evaluate how TETRAD algorithms perform on data generated using other software packages. **algcomparison** will save data and graphs to disk in a common format. These data and graphs can be loaded and analyzed in other platforms, such as R, MATLAB, or Python. The results of these analyses can then be stored, read by **algcomparison**, and included in comparison tables alongside TETRAD results. Data simulated on other platforms may also be loaded into the **algcomparison** tool. Instructions for executing cross-platform comparison are found on the project page: `https://github.com/bd2kccd/causal-compare`.

4. The code for the command line interface is separate from TETRAD and can be found here: `https://github.com/bd2kccd/causal-compare`.

Acknowledgments

Research reported in this publication was supported by grant U54HG008540 awarded by the National Human Genome Research Institute through funds provided by the trans-NIH Big Data to Knowledge (BD2K) initiative.

References

David A Bessler and Seongpyo Lee. Money and prices: US data 1869–1914 (a study with directed graphs). *Empirical Economics*, 27(3):427–446, 2002.

Selva Demiralp and Kevin D Hoover. Searching for the causal structure of a vector autoregression. *Oxford Bulletin of Economics and Statistics*, 65(s1):745–767, 2003.

Mathias Drton and Marloes H Maathuis. Structure learning in graphical modeling. *Annual Review of Statistics and Its Application*, 4:365–393, 2017.

Imme Ebert-Uphoff and Yi Deng. Causal discovery for climate research using graphical models. *Journal of Climate*, 25(17):5648–5665, 2012.

Clark Glymour, Richard Scheines, Peter Spirtes, and Kevin Kelly. *Discovering Causal Structure: Artificial Intelligence, Philosophy of Science, and Statistical Modeling.* Academic Press, 1987.

Christina Heinze-Deml, Marloes H Maathuis, and Nicolai Meinshausen. Causal structure learning. *Annual Review of Statistics and Its Application*, 5:371–391, 2018.

Markus Kalisch, Martin Mächler, Diego Colombo, Marloes H Maathuis, and Peter Bühlmann. Causal inference using graphical models with the R package **pcalg**. *Journal of Statistical Software*, 47(11):1–26, 2012.

Colleen Mills-Finnerty, Catherine Hanson, and Stephen Jose Hanson. Brain network response underlying decisions about abstract reinforcers. *NeuroImage*, 103:48–54, 2014.

Kevin Murphy. The Bayes Net Toolbox for Matlab. *Computing Science and Statistics*, 33 (2):1024–1034, 2001.

Martina Rau, Richard Scheines, Vincent Aleven, and Nikol Rummel. Does representational understanding enhance fluency – or vice versa? Searching for mediation models. In *Proceedings of the Sixth International Conference on Educational Data Mining*, 2013.

Richard Scheines, Peter Spirtes, Clark Glymour, Christopher Meek, and Thomas Richardson. The TETRAD project: Constraint based aids to causal model specification. *Multivariate Behavioral Research*, 33(1):65–117, 1998.

Marco Scutari. Learning bayesian networks with the **bnlearn** R package. *Journal of Statistical Software*, 35(3):1–22, 2010.

Bill Shipley, Martin J Lechowicz, Ian Wright, and Peter B Reich. Fundamental trade-offs generating the worldwide leaf economics spectrum. *Ecology*, 87(3):535–541, 2006.

Stephen M Smith, Karla L Miller, Gholamreza Salimi-Khorshidi, Matthew Webster, Christian F Beckmann, Thomas E Nichols, Joseph D Ramsey, and Mark W Woolrich. Network modelling methods for fMRI. *NeuroImage*, 54(2):875–891, 2011.

Peter Spirtes and Kun Zhang. Causal discovery and inference: concepts and recent methodological advances. *Applied Informatics*, 3(3), 2016.

Peter Spirtes, Richard Scheines, and Clark Glymour. Simulation studies of the reliability of computer-aided model specification using the TETRAD II, EQS, and LISREL programs. *Sociological Methods & Research*, 19(1):3–66, 1990.

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