Journal of Machine Learning Research 25 (2024) 1-8 Submitted 4/23; Revised 11/23; Published 9/24

pgmpy: A Python Toolkit for Bayesian Networks

Ankur Ankan ANKUR.ANKAN@RU.NL

Johannes Textor JOHANNES.TEXTOR@RU.NL

Institute of Computing and Information Sciences, Radboud University, Nijmegen, Netherlands

Editor: Antti Honkela

Abstract

Bayesian Networks (BNs) are used in various fields for modeling, prediction, and decision making. pgmpy is a python package that provides a collection of algorithms and tools to work with BNs and related models. It implements algorithms for structure learning, parameter estimation, approximate and exact inference, causal inference, and simulations. These implementations focus on modularity and easy extensibility to allow users to quickly modify/add to existing algorithms, or to implement new algorithms for different use cases. pgmpy is released under the MIT License; the source code is available at: `https://github.com/pgmpy/pgmpy`, and the documentation at: `https://pgmpy.org`.

Keywords: Bayesian Networks, Directed Acyclic Graphs, Causal Inference, Probabilistic Inference, Simulation, Structure Learning, Causal Discovery

1. Introduction

Bayesian Networks (BNs), also known as Belief Networks, and related models such as Directed Acyclic Graphs (DAGs), Structural Equation Models (SEMs), and Dynamic Bayesian Networks (DBNs) are used in a variety of applications (Pourret et al., 2008) like healthcare (Kyrimi et al., 2021), medicine (Arora et al., 2019), natural language processing (Goyal et al., 2008), computational biology (Needham et al., 2007; Chen et al., 2022), robotics (Lazkano et al., 2007; Premebida et al., 2016; Amiri et al., 2022), neuroscience (Bielza and Larrañaga, 2014), and others. Some of the most common tasks in these applications are learning model structure from data (structure learning), estimating model parameters (parameter learning), querying a model for conditional or marginal distributions (probabilistic inference), causal effect estimation between variables (causal inference), and simulating data under various generating processes. pgmpy provides algorithms to perform all these tasks for discrete variable BNs, with a subset of algorithms supporting continuous and mixed data. We describe the main features and the implemented algorithms in detail in Section 2.

As numerous algorithms for solving BN tasks are continuously being developed, a primary design objective of pgmpy is to faciliate modifying or adding to the implemented algorithms and enabling users to quickly implement new ones. To achieve this, pgmpy is written in pure python for readability with a focus on code modularity. Most algorithm implementations accept custom user-defined functions and classes as arguments. Each class of algorithms conform to their abstract base class, enabling users to quickly implement new algorithms by inheriting these base classes and using them seamlessly with other features.

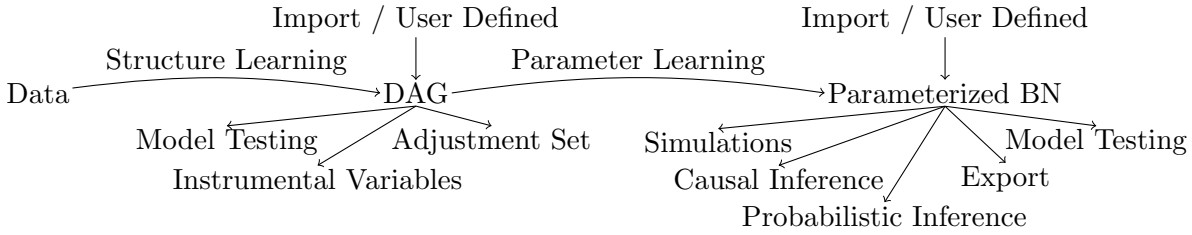

Figure 1: Possible workflows in pgmpy for Bayesian Networks

In addition to extensibility, pgmpy ensures scalability of the implemented algorithms by benchmarking runtime and memory usage (Ankan, 2023), and reliability by extensive unit testing and validation against other open-source packages.

2. Main Features

This section gives an overview of the main features and the implemented algorithms. The algorithms only support discrete variables unless stated otherwise.

Structure Learning (SL): Given a dataset, SL algorithms estimate the model structure. pgmpy implements three general-purpose SL algorithms:

1. **PC:** A constraint-based algorithm that exploits the conditional independences (CIs) in the data to construct the model structure. Three variants of the algorithm are implemented: a) Original (Spirtes et al., 1993), b) Stable (Colombo et al., 2014), and c) Parallel (Le et al., 2019). The following CI tests are available: a) chi-squared, b) G-test, c) Cressie-Read Power-Divergence, d) partial correlation, and e) residualization test (Ankan and Textor, 2023). The partial correlation test can be used only for continuous data and the residualization test can be used for mixed data. Users can also specify custom tests as a function.
2. **Hill-Climb Search (HC):** A score-based algorithm that iteratively makes local changes in the model structure to improve its score. The following scoring metrics are implemented: a) K2 Score (Cooper and Herskovits, 1992), b) BDeu (Buntine, 1991), c) BDs (Scutari, 2016), d) BIC (Schwarz, 1978), e) AIC (Akaike, 1974). Users can also specify custom scoring methods.
3. **Max-Min Hill-Climbing(MMHC):** The MMHC algorithm (Tsamardinos et al., 2006) combines constraint-based and score-based approaches by using the output of the MMPC algorithm (Tsamardinos et al., 2003), which is a constraint-based algorithm, as the starting point for HC algorithm.

Apart from the general purpose SL algorithms, pgmpy also implements the Chow-Liu algorithm (Chow and Liu, 1968) and Tree Augmented Naive Bayes (Friedman et al., 1997) to learn tree structures. Both algorithms are based on constructing a maximum spanning tree over the variables using a (conditional) mutual information based metric as the edge weights. Users can use mutual information (or its variants like adjusted or normalized mutual information) as the edge weights or they can specify a custom edge weight function.

Parameter Learning: The SL algorithms output a DAG, which can be used for causal inference analyses like finding adjustment sets or instrumental variables (Figure 1). However,

for applications like inference and simulations we also need to learn the model's parameters. The following three methods are implemented to estimate the Conditional Probability Distribution (CPD), also known as Conditional Probability Tables (CPTs), for the variables from (weighted) data:

1. **Maximum Likelihood (ML) Estimator:** Learns the ML estimates for the CPDs.
2. **Bayesian Estimator:** Uses user-specified priors such as Dirichlet, K2, BDeu or user-defined, for each variable to perform a Bayesian estimate of the CPDs.
3. **Expectation Maximization (EM):** Uses the EM algorithm to make Maximum Likelihood estimates in the presence of latent variables or missing data.

Probabilistic Inference: Given a fully specified BN, the probabilistic inference algorithms allow users to query the model for any conditional distribution, $P(\boldsymbol{X}|\boldsymbol{Y} = \boldsymbol{Y_i})$. Currently, two algorithms are implemented for exact inference, and a simulation-based approach is implemented for approximate inference. All these algorithms allow users to specify combinations of hard and virtual (Pearl, 1988) evidence.

1. **Variable Elimination (VE):** The VE algorithm (Zhang and Poole, 1994) conditions the model on the given evidence and iteratively sums out all the variables that are not in the query. The order in which the variables are summed out is crucial for the computational cost. pgmpy implements some algorithms and heuristics (Koller and Friedman, 2009) to compute an efficient elimination order: 1) Optimized einsum-based (Smith and Gray, 2018) 2) Min Fill 3) Min Neighbours 4) Min Weight 5) Weighted Min Fill. Users can also specify a custom algorithm or a list of variable names.
2. **Belief Propagation (BP):** The BP algorithm (Pearl, 1982) splits the query computation into two parts – model calibration and query computation. It can be much faster than VE when multiple queries need to be made on the same model, as the calibration step does not need to be repeated.
3. **Approximate Inference**: Depending on the model size and the query, both VE and BP can become intractable. In such cases, approximate inference can be used which computes the queried distribution by simulating data from the model.

Causal Inference: The causal inference module provides features to estimate the causal effect between a given exposure and an outcome variable. This can be done using:

1. **Instrumental Variables (IVs):** This method finds IVs and conditional IVs (Van der Zander et al., 2015) that can be used with an IV-based estimator to get the estimates.
2. **Adjustment Set:** This method finds the minimal adjustment set (Perković et al., 2018) that can be used with any statistical model to get the estimates.

For continuous datasets, pgmpy implements a Two-Stage Least Squares (2SLS) estimator for the IV method and a linear regression model for the adjustment set method. For discrete datasets, estimates are computed using the probabilistic inference engine. Users can also choose to use pgmpy to find the IVs and adjustment sets, and use other statistical models for estimation using packages such as statsmodels (Seabold and Perktold, 2010)).

Model Testing: The model testing module provides methods to compare the fit of different models to a given dataset. The following three metrics are implemented: 1) Log-likelihood Score: Computes the log-likelihood of the dataset given a fully-specified BN. 2) Structure Score: Computes the structure score of a given model using one of the scoring

Package	SL	PL	EI	AI	Sim	CaI	cSim	MT	Ex/Im
pgmpy	✓	✓	✓	✓	✓	✓	✓	✓	✓
pomegranate (Schreiber, 2017)	✓	✓		✓	✓				
pyBNesian (Atienza et al., 2022)	✓	✓			✓				
bayespy (Luttinen, 2016)		✓		✓					
DoWhy (Blöbaum et al., 2022)	✓					✓		✓	
dagitty (Textor et al., 2016)					✓	✓		✓	✓
bnlearn (Scutari, 2010)	✓	✓	✓	✓	✓				✓
pcalg (Kalisch et al., 2012)	✓	✓	✓	✓	✓	✓			✓
libDAI (Mooij, 2010)			✓	✓					

Table 1: Feature comparison with other open-source packages. causalnex (Beaumont et al., 2021) and python bnlearn (Taskesen, 2020) have been excluded as they both extend pgmpy. SL=Structure Learning, PL=Parameter Learning, EI=Exact Inference, AI=Approximate Inference, Sim=Simulation, CaI=Causal Inference, cSim=Causal Simulation, MT=Model Testing, Ex/Im=Export/Import

metrics for the HC algorithm. 3) Correlation Score (Ankan and Textor, 2023): Aims to help in understanding how well the model explains the correlations present in the dataset by comparing the d-connected variables with the correlated variables in the dataset.

Simulations: The simulation method allows users to generate data from a fully specified BN under any combination of the following conditions: a) Hard evidence, b) Virtual evidence (Pearl, 1988), c) Hard intervention, and d) Virtual intervention.

Exporting and Importing models: Finally, to interface with other packages pgmpy supports importing/exporting models in the following file formats: a) Bayesian Interchange Format (BIF) (Hulten and Domingos, 2003), b) UAI (Noetic Systems, 2022), c) XMLBIF (Cozman et al., 1998), d) XMLBeliefNetwork (Microsoft Research, 1999), and e) NET (HUGIN EXPERT A/S , 2011).

3. Conclusion and Future Work

pgmpy provides a collection of tools and algorithms to work with BNs and related models with a design focus on modularity and extensibility. While there are other python and R packages offering similar features, most are specialized for either probabilistic inference tasks or causal inference tasks (Table 1). In contrast, pgmpy combines these and allows users to use methods from both these fields seamlessly.

In the future, we plan to add an alternative logarithmic scale computational backend to avoid underflow and improve numerical accuracy, expand support for continuous variables, and introduce more algorithms for approximate inference, such as Gibbs Sampling and Loopy Belief Propagation. Furthermore, we aim to expand causal inference features by adding new model classes, such as Maximal Ancestral Graphs and Partial Ancestral Graphs, and by implementing causal discovery algorithms like Fast Causal Inference and Greedy Equivalence Search.

Acknowledgements

We would like to thank all the contributors of pgmpy. We would also like to thank Google and Python Software Foundation for their support through the Google Summer of Code program.

References

H. Akaike. A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19(6):716–723, December 1974. doi: 10.1109/tac.1974.1100705. URL https://doi.org/10.1109/tac.1974.1100705.

Saeid Amiri, Kishan Chandan, and Shiqi Zhang. Reasoning with scene graphs for robot planning under partial observability. *IEEE Robotics and Automation Letters*, 7(2):5560–5567, April 2022. doi: 10.1109/lra.2022.3157567. URL https://doi.org/10.1109/lra.2022.3157567.

Ankur Ankan. pgmpy runtime and memory benchmarks. https://pgmpy.org/pgmpy-benchmarks, 2023. [Online; Accessed November 3, 2023].

Ankur Ankan and Johannes Textor. A simple unified approach to testing high-dimensional conditional independences for categorical and ordinal data. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(10):12180–12188, Jun. 2023. doi: 10.1609/aaai.v37i10.26436. URL https://ojs.aaai.org/index.php/AAAI/article/view/26436.

Paul Arora, Devon Boyne, Justin J. Slater, Alind Gupta, Darren R. Brenner, and Marek J. Druzdzel. Bayesian Networks for risk prediction using real-world data: A tool for precision medicine. *Value in Health*, 22(4):439–445, April 2019. doi: 10.1016/j.jval.2019.01.006. URL https://doi.org/10.1016/j.jval.2019.01.006.

David Atienza, Concha Bielza, and Pedro Larrañaga. PyBNesian: an extensible python package for Bayesian Networks. *Neurocomputing*, 504:204–209, September 2022. doi: 10.1016/j.neucom.2022.06.112. URL https://doi.org/10.1016/j.neucom.2022.06.112.

Paul Beaumont, Ben Horsburgh, Philip Pilgerstorfer, Angel Droth, Richard Oentaryo, Steven Ler, Hiep Nguyen, Gabriel Azevedo Ferreira, Zain Patel, and Wesley Leong. CausalNex, 10 2021. URL https://github.com/quantumblacklabs/causalnex.

Concha Bielza and Pedro Larrañaga. Bayesian Networks in neuroscience: a survey. *Frontiers in Computational Neuroscience*, 8, October 2014. doi: 10.3389/fncom.2014.00131. URL https://doi.org/10.3389/fncom.2014.00131.

Patrick Blöbaum, Peter Götz, Kailash Budhathoki, Atalanti A. Mastakouri, and Dominik Janzing. Dowhy-gcm: An extension of dowhy for causal inference in graphical causal models. *arXiv preprint arXiv:2206.06821*, 2022.

Wray Buntine. Theory refinement on Bayesian Networks. In *Uncertainty Proceedings 1991*, pages 52–60. Elsevier, 1991. doi: 10.1016/b978-1-55860-203-8.50010-3. URL https://doi.org/10.1016/b978-1-55860-203-8.50010-3.

Xueer Chen, Lujia Chen, Cornelius H. L. Kürten, Fattaneh Jabbari, Lazar Vujanovic, Ying Ding, Binfeng Lu, Kevin Lu, Aditi Kulkarni, Tracy Tabib, Robert Lafyatis, Gregory F. Cooper, Robert Ferris, and Xinghua Lu. An individualized causal framework for learning intercellular communication networks that define microenvironments of individual tumors. *PLOS Computational Biology*, 18(12):e1010761, December 2022. doi: 10.1371/journal.pcbi.1010761. URL https://doi.org/10.1371/journal.pcbi.1010761.

C. Chow and C. Liu. Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, 14(3):462–467, May 1968. doi: 10.1109/tit.1968.1054142. URL https://doi.org/10.1109/tit.1968.1054142.

Diego Colombo, Marloes H Maathuis, et al. Order-independent constraint-based causal structure learning. *Journal of Machine Learning Research*, 15(1):3741–3782, 2014.

Gregory F. Cooper and Edward Herskovits. A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9(4):309–347, October 1992. doi: 10.1007/bf00994110. URL https://doi.org/10.1007/bf00994110.

Fabio Cozman, Marek Drunzdzel, and Garcia Daniel. The interchange format for Bayesian Networks, 1998. URL https://www.cs.cmu.edu/afs/cs/user/fgcozman/www/Research/InterchangeFormat/. [Accessed 14-Apr-2023].

Nir Friedman, Dan Geiger, and Moises Goldszmidt. Bayesian Network classifiers, 1997. URL https://doi.org/10.1023/a:1007465528199.

Pawan Goyal, Laxmidhar Behera, and TM McGinnity. Application of Bayesian framework in natural language understanding. *IETE Technical Review*, 25(5):251, 2008. doi: 10.4103/0256-4602.44656. URL https://doi.org/10.4103/0256-4602.44656.

HUGIN EXPERT A/S . The HUGIN file format, 2011. URL http://www.hugin.com. [Accessed 14-Apr-2023].

Geoff Hulten and Pedro Domingos. The interchange format for Bayesian Networks, 2003. URL http://www.cs.washington.edu/dm/vfml/appendixes/bif.htm. [Accessed 14-Apr-2023].

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), 2012. doi: 10.18637/jss.v047.i11. URL https://doi.org/10.18637/jss.v047.i11.

Daphne Koller and Nir Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT press, 2009.

Evangelia Kyrimi, Scott McLachlan, Kudakwashe Dube, Mariana R. Neves, Ali Fahmi, and Norman Fenton. A comprehensive scoping review of Bayesian Networks in healthcare: Past, present and future. *Artificial Intelligence in Medicine*, 117:102108, July 2021. doi: 10.1016/j.artmed.2021.102108. URL https://doi.org/10.1016/j.artmed.2021.102108.

E. Lazkano, B. Sierra, A. Astigarraga, and J.M. Martínez-Otzeta. On the use of Bayesian Networks to develop behaviours for mobile robots. *Robotics and Autonomous Systems*, 55(3):253–265, March 2007. doi: 10.1016/j.robot.2006.08.003. URL https://doi.org/10.1016/j.robot.2006.08.003.

Thuc Duy Le, Tao Hoang, Jiuyong Li, Lin Liu, Huawen Liu, and Shu Hu. A fast PC algorithm for high dimensional causal discovery with multi-core PCs. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 16(5):1483–1495, September 2019. doi: 10.1109/tcbb.2016.2591526. URL https://doi.org/10.1109/tcbb.2016.2591526.

Jaakko Luttinen. Bayespy: variational Bayesian inference in python. *The Journal of Machine Learning Research*, 17(1):1419–1424, 2016.

Microsoft Research. XML belief network file format, 1999. URL http://xml.coverpages.org/xbn-MSdefault19990414.html. [Accessed 14-Apr-2023].

Joris M. Mooij. libdai: A free and open source C++ library for discrete approximate inference in graphical models. *Journal of Machine Learning Research*, 11(74):2169–2173, 2010. URL http://jmlr.org/papers/v11/mooij10a.html.

Chris J Needham, James R Bradford, Andrew J Bulpitt, and David R Westhead. A primer on learning in Bayesian Networks for computational biology. *PLoS Computational Biology*, 3(8):e129, August 2007. doi: 10.1371/journal.pcbi.0030129. URL https://doi.org/10.1371/journal.pcbi.0030129.

Noetic Systems. UAI file format, 2022. URL https://forgemia.inra.fr/thomas.schiex/toulbar2/-/blob/master/doc/UAI08Format.txt. [Accessed 14-Apr-2023].

Judea Pearl. Reverend Bayes on inference engines: A distributed hierarchical approach. In *Proceedings of the Second AAAI Conference on Artificial Intelligence*, AAAI'82, page 133–136. AAAI Press, 1982.

Judea Pearl. *Probabilistic Reasoning in Intelligent Systems*. Elsevier, 1988. doi: 10.1016/c2009-0-27609-4. URL https://doi.org/10.1016/c2009-0-27609-4.

Emilija Perković, Johannes Textor, Markus Kalisch, and Marloes H. Maathuis. Complete graphical characterization and construction of adjustment sets in markov equivalence classes of ancestral graphs. *Journal of Machine Learning Research*, 18(220):1–62, 2018. URL http://jmlr.org/papers/v18/16-319.html.

Olivier Pourret, Patrick Na, Bruce Marcot, et al. *Bayesian Networks: A Practical Guide to Applications*. John Wiley & Sons, 2008.

Cristiano Premebida, Diego R. Faria, and Urbano Nunes. Dynamic Bayesian Network for semantic place classification in mobile robotics. *Autonomous Robots*, 41(5):1161–1172, July 2016. doi: 10.1007/s10514-016-9600-2. URL https://doi.org/10.1007/s10514-016-9600-2.

Jacob Schreiber. Pomegranate: fast and flexible probabilistic modeling in python. *The Journal of Machine Learning Research*, 18(1):5992–5997, 2017.

Gideon Schwarz. Estimating the dimension of a model. *The Annals of Statistics*, 6(2), March 1978. doi: 10.1214/aos/1176344136. URL https://doi.org/10.1214/aos/1176344136.

Marco Scutari. Learning Bayesian Networks with the bnlearn R package. *Journal of Statistical Software*, 35(3), 2010. doi: 10.18637/jss.v035.i03. URL https://doi.org/10.18637/jss.v035.i03.

Marco Scutari. An empirical Bayes score for discrete Bayesian Networks. In *Conference on probabilistic graphical models*, pages 438–448. PMLR, 2016.

Skipper Seabold and Josef Perktold. statsmodels: Econometric and statistical modeling with python. In *9th Python in Science Conference*, 2010.

Daniel Smith and Johnnie Gray. opt_einsum - a python package for optimizing contraction order for einsum-like expressions. *Journal of Open Source Software*, 3(26):753, 2018. doi: 10.21105/joss.00753. URL https://doi.org/10.21105/joss.00753.

Peter Spirtes, Clark Glymour, and Richard Scheines. *Causation, Prediction, and Search*. Springer New York, 1993. doi: 10.1007/978-1-4612-2748-9. URL https://doi.org/10.1007/978-1-4612-2748-9.

Erdogan Taskesen. bnlearn - library for Bayesian Network learning and inference, 1 2020. URL https://erdogant.github.io/bnlearn.

Johannes Textor, Benito Van der Zander, Mark S Gilthorpe, Maciej Liśkiewicz, and George TH Ellison. Robust causal inference using directed acyclic graphs: the R package 'dagitty'. *International Journal of Epidemiology (IJE)*, 45(6):1887–1894, 2016.

Ioannis Tsamardinos, Constantin F. Aliferis, and Alexander Statnikov. Time and sample efficient discovery of Markov blankets and direct causal relations. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, August 2003. doi: 10.1145/956750.956838. URL https://doi.org/10.1145/956750.956838.

Ioannis Tsamardinos, Laura E. Brown, and Constantin F. Aliferis. The max-min hill-climbing Bayesian Network structure learning algorithm. *Machine Learning*, 65(1):31–78, March 2006. doi: 10.1007/s10994-006-6889-7. URL https://doi.org/10.1007/s10994-006-6889-7.

Benito Van der Zander, Johannes Textor, and Maciej Liskiewicz. Efficiently finding conditional instruments for causal inference. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

Nevin L Zhang and David Poole. A simple approach to Bayesian Network computations. In *Proc. of the Tenth Canadian Conference on Artificial Intelligence*, 1994.