Journal of Machine Learning Research 10 (2009) 159-162 Submitted 6/08; Revised 10/08; Published 2/09

Python Environment for Bayesian Learning: Inferring the Structure of Bayesian Networks from Knowledge and Data

Abhik Shah SHAHAD@UMICH.EDU
Peter Woolf PWOOLF@UMICH.EDU
Department of Chemical Engineering
3320 G.G. Brown
Ann Arbor, MI 48103, USA

Editor: Cheng Soon Ong

Abstract

In this paper, we introduce PEBL, a Python library and application for learning Bayesian network structure from data and prior knowledge that provides features unmatched by alternative software packages: the ability to use interventional data, flexible specification of structural priors, modeling with hidden variables and exploitation of parallel processing.
PEBL is released under the MIT open-source license, can be installed from the Python Package Index and is available at `http://pebl-project.googlecode.com`.

Keywords: Bayesian networks, python, open source software

1. Introduction

Bayesian networks (BN) have become a popular methodology in many fields because they can model nonlinear, multimodal relationships using noisy, inconsistent data. Although learning the structure of BNs from data is now common, there is still a great need for high-quality open-source software that can meet the needs of various users. End users require software that is easy to use; supports learning with different data types; can accommodate missing values and hidden variables; and can take advantage of various computational clusters and grids. Researchers require a framework for developing and testing new algorithms and translating them into usable software. We have developed the Python Environment for Bayesian Learning (PEBL) to meet these needs.

2. PEBL Features

PEBL provides many features for working with data and BNs; some of the more notable ones are listed below.

2.1 Structure Learning

PEBL can load data from tab-delimited text files with continuous, discrete and class variables and can perform maximum entropy discretization. Data collected following an intervention is important for determining causality but requires an altered scoring procedure (Pe'er et al., 2001; Sachs et al., 2002). PEBL uses the BDe metric for scoring networks and handles interventional data using the method described by Yoo et al. (2002).

```
from pebl import data, result
from pebl.learner.greedy import GreedyLearner
from pebl.taskcontroller.xgrid import XgridController

dataset = data.fromfile('mydata.txt')
runner = XgridController('grid.com', 'pass')
learners = [GreedyLearner(dataset) for i in range(10)]
results = runner.run(learners)
result.merge(results).tohtml('./myoutput')
```

(a) Python script

```
[data]
filename = mydata.txt
[learner]
type = greedy.GreedyLearner
numtasks = 10
[taskcontroller]
type=xgrid.XgridController
[xgrid]
controller = grid.com
password = pass
[result]
output = ./myoutput
```

(b) PEBL configuration file

Figure 1: Two ways of using PEBL: with a Python script and a configuration file. Both methods create 10 greedy learners with default parameters and run them on an Apple Xgid. The Python script can be typed in an interactive shell, run as a script or included as part of a larger application.

PEBL can handle missing values and hidden variables using exact marginalization and Gibbs sampling (Heckerman, 1998). The Gibbs sampler can be resumed from a previously suspended state, allowing for interactive inspection of preliminary results or a manual strategy for determining satisfactory convergence.

A key strength of Bayesian analysis is the ability to use prior knowledge. PEBL supports structural priors over edges specified as 'hard' constraints or 'soft' energy matrices (Imoto et al., 2003) and arbitrary constraints specified as Python functions or lambda expressions.

PEBL includes greedy hill-climbing and simulated annealing learners and makes writing custom learners easy. Efficient implementaion of learners requires careful programming to eliminate redundant computation. PEBL provides components to alter, score and rollback changes to BNs in a simple, transactional manner and with these, efficient learners look remarkably similar to pseudocode.

2.2 Convenience and Scalability

PEBL includes both a library and a command line application. It aims for a balance between ease of use, extensibility and performance. The majority of PEBL is written in Python, a dynamically-typed programming language that runs on all major operating systems. Critical sections use the numpy library (Ascher et al., 2001) for high-performance matrix operations and custom extensions written in ANSI C for portability and speed.

PEBL's use of Python makes it suitable for both programmers and domain experts. Python provides interactive shells and notebook interfaces and includes an extensive standard library and many third-party packages. It has a strong presence in the scientific computing community (Oliphant, 2007). Figure 1 shows a script and configuration file example that showcase the ease of using PEBL.

	BANJO	**BNT**	**Causal Explorer**	**Deal**	**LibB**	**PEBL**
Latest Version	2.0.1	1.04	1.4	1.2-25	2.1	0.9.10
License	Academic [1]	GPL	Academic [1]	GPL	Academic [1]	MIT
Scripting Language	Matlab [2]	Matlab	Matlab	R	N/A	Python
Application	Yes	No	No	No	Yes	Yes
Interventional Data	No	Yes	No	No	No	Yes
DBN	Yes	Yes	No	No	No	No
Structural Priors	Yes [3]	No	No	No	No	Yes
Missing Data	No	Yes	No	No	Yes	Yes
Parallel Execution	No	No	No	No	No	Yes

[1] Custom academic, non-commercial license; not OSI approved.
[2] Via a Matlab-Java bridge.
[3] Only constraints/hard-priors supported.

Table 1: Comparing the features of popular Bayesian network structure learning software.

While many tasks related to Bayesian learning are embarrassingly parallel in theory, few software packages take advantage of it. PEBL can execute learning tasks in parallel over multiple processors or CPU cores, an Apple Xgrid,[1] an IPython cluster[2] or the Amazon EC2 platform.[3] The EC2 platform is especially attractive for scientists because it allows one to rent processing power on an on-demand basis and execute PEBL tasks on them.

With appropriate configuration settings and the use of parallel execution, PEBL can be used for large learning tasks. Although PEBL has been tested successfully with data sets with 10000 variables and samples, BN structure learning is a known NP-Hard problem (Chickering et al., 1994) and analysis using data sets with more than a few hundred variables is likely to result in poor results due to poor coverage of the search space.

3. PEBL Development

The benefits of open source software derive not just from the freedoms afforded by the software license but also from the open and collaborative development model. PEBL's source code repository and issue tracker are hosted at Google Code and freely available to all. Additionally, PEBL includes over 200 automated unit tests and mandates that every source code submission and resolved error be accompanied with tests.

4. Related Software

While there are many software tools for working with BNs, most focus on parameter learning and inference rather than structure learning. Of the few tools for structure learning, few are open-source and none provide the set of features included in PEBL. As shown in Table 1, the ability to handle interventional data, model with missing values and hidden variables, use soft and arbitrary priors and exploit parallel platforms are unique to PEBL. PEBL, however, does not currently provide any

1. Grid computing solution by Apple, Inc. `http://www.apple.com/server/macosx/technology/xgrid.html`.
2. Cluster of Python interpreters (`http://ipython.scipy.org`).
3. A pay-per-use, on-demand computing platform by Amazon, Inc. (`http://aws.amazon.com`).

features for inference or learning Dynamic Bayesian Networks (DBN). Despite its use of optimized matrix libraries and custom C extension modules, PEBL can be an order of magnitude or more slower than software written in Java or C/C++; the ability to use a wider range of data and priors, the parallel processing features and the ease-of-use, however, should make it an attractive option for many users.

5. Conclusion and Future Work

We have developed a library and application for learning BNs from data and prior knowledge. The set of features found in PEBL is unmatched by alternative packages and we hope that our open development model will convince others to use PEBL as a platform for BN algorithms research.

Acknowledgments

We would like to acknowledge support for this project from the NIH grant #U54 DA021519.

References

D. Ascher, P. F. Dubois, K. Hinsen, J. Hugunin, and T. Oliphant. An open source project: Numerical python. Technical Report UCRL-MA-128569, Lawrence Livermore National Laboratory, September 2001.

D. M. Chickering, D. Geiger, and D. Heckerman. Learning bayesian networks is np-hard. Technical Report MSR-TR-94-17, Microsoft Research, November 1994.

D. Heckerman. A tutorial on learning with bayesian networks. In *Learning in Graphical Models*, pages 301–354. The MIT Press, 1998.

S. Imoto, T. Higuchi, T. Goto, K. Tashiro, S. Kuhara, and S. Miyano. Combining microarrays and biological knowledge for estimating gene networks via bayesian networks. *Bioinformatics Conference, 2003. CSB 2003. Proceedings of the 2003 IEEE*, pages 104–113, 2003.

T. E. Oliphant. Python for scientific computing. *Computing in Science & Engineering*, pages 10–20, 2007.

D. Pe'er, A. Regev, G. Elidan, and N. Friedman. Inferring subnetworks from perturbed expression profiles. *Bioinformatics*, 1(1):1–9, 2001.

K. Sachs, D. Gifford, T. Jaakkola, P. Sorger, and D. Lauffenburger. Bayesian network approach to cell signaling pathway modeling. *Science's STKE*, 2002.

C. Yoo, V. Thorsson, and G. F. Cooper. Discovery of causal relationships in a gene-regulation pathway from a mixture of experimental and observational DNA microarray data. *Pac Symp Biocomput*, 7:498–509, 2002.