aGrUM/pyAgrum : a toolbox to build models and algorithms for Probabilistic Graphical Models in Python

Gaspard Ducamp, Christophe Gonzales, Pierre-Henri Wuillemin

aGrUM/pyAgrum : a Toolbox to Build Models and Algorithms for Probabilistic Graphical Models in Python

Gaspard Ducamp GASPARD.DUCAMP@LIP6.FR
LIP6 - Sorbonne University
4, place Jussieu, 75005, Paris France

Christophe Gonzales CHRISTOPHE.GONZALES@LIS-LAB.FR
Aix Marseille Univ, Université de Toulon, CNRS, LIS, Marseille, France

Pierre-Henri Wuillemin PIERRE-HENRI.WUILLEMIN@LIP6.FR
LIP6 - Sorbonne University
4, place Jussieu, 75005, Paris France

Abstract

This paper presents the aGrUM framework, a LGPL C++ library providing state-of-the-art implementations of graphical models for decision making, including Bayesian Networks, Markov Networks (Markov random fields), Influence Diagrams, Credal Networks, Probabilistic Relational Models. The framework also contains a wrapper, pyAgrum for exploiting aGrUM in Python. This framework is the result of an ongoing effort to build an efficient and well maintained open source cross-platform software, running on Linux, MacOS X and Windows, for dealing with graphical models and for providing essential components to build new algorithms for graphical models.

Keywords: Bayesian Networks, Probabilistic Graphical Models, c++, python

1. Introduction

Graphical models (GM) are a core technology in AI. In this context, the aGrUM/pyAgrum framework provides efficient, effective, easy-to-use and well maintained tools to deal with graphical models. The emphasis is set on high standards for performance, code quality and usability. As such, the aGrUM part is written in C++ and is in charge of the computation intensive tasks whereas pyAgrum, a Python wrapper, is more focused on user-friendly interactions and delegates most of the "hard" work to aGrUM. All the code runs on Linux, MacOS and Windows operating systems (supported compilers include g++, clang, mvsc, mingw). The aGrUM/pyAgrum project started twelve years ago at the artificial intelligence and decision department of University Pierre and Marie Curie (http://www.lip6.fr) and has now become an extensive open source graphical model framework under LGPL licence. It is freely available at https://agrum.gitlab.io.

2. aGrUM's Features

The aGrUM C++ library is divided into eight modules, the majority of which are related to different graphical models:

- BN: Bayesian Networks.
- MN: Markov Networks (Markov Random Field).
- CN: Credal Networks
- FMDP: Factorized Markov Decision Processes
- ID: Influence Diagrams.
- PRM: Probabilistic Relational Models
- Learning: Bayesian Network structure and parameters learning algorithms
- Core: data structures and utilities useful for the other modules.

The BN module provides flexible and efficient implementations to handle Bayesian Networks. Those can be created by specifying Conditional Probability Tables, Noisy OR or Noisy AND gates, Logit models, aggregators (and, or, max, min, exists, forall, *etc.*). They can also be learnt from data (see below). BNs can be read from and written to files of different formats (BIF, DSL, net, cnf, BIFXML, UAI). The library also provides efficient approximate and exact inference algorithms (Gibbs and importance sampling, loopy belief, lazy propagation, Shafer-Shenoy, variable elimination, *etc.*), some of them having incremental features and exploiting relevant reasoning to speed-up computations.

A specific module is devoted to learning the structure and/or parameters of BNs from datasets (CSV files or SQL databases). To be as flexible as possible, it exploits a component-based approach: every high-level learning algorithm combines i) a constraint component (e.g., requiring/forbidding arcs, limiting the indegrees, imposing a partial node ordering, *etc.*), ii) a selector determining which graphical structures are allowed (e.g., taking into account the constraints), iii) a handler to parse all or only part of datasets, thereby providing very effective parallelizations (using either OpenMP or the C++ thread library) and easy cross-validation schemes. Of course, the library includes the classical structure learning scores (BD, BDeu, K2, AIC, BIC/MDL, fNML) but also various *a priori* (smoothing, Dirichlet). The latter are also taken into account during parameter learning. The currently implemented high-level learning algorithms are greedy hill climbing, local search with tabu list, K2, 3off2 and MIIC.

Besides BNs, other GMs have been implemented: Markov Random Field (module MN), Credal Networks (CN), Factorized Markov Decision Processes (FMDP), Influence Diagrams (ID) and Probabilistic Relational Models (PRM). They follow the same philosophy as the BN module: high flexibility, inference efficiency, extended file format support. For instance, all these models are shipped with tailored inference algorithms, e.g., loopy propagation and Monte Carlo for CN, SPUDD for FMDPs, Shafer-Shenoy for MNs and IDs.

All the aforementioned modules rely on the core module for their data structures and common algorithms. These include tailored implementations of classical data structures like hashtables, sets, heaps, *etc.*, but also mathematical functions (chi2, GammaLog2 functions, *etc.*) and graph definitions and algorithms (e.g., a whole hierarchy of triangulations, notably incremental ones) and different flavors of multidimensional tables (tensors, sparse matrices, aggregators, *etc.*). The core of the `aGrUM` library also contains some tools used to make sure that `aGrUM`'s code satisfies the highest quality standards and is memory leak free.

3. pyAgrum

Beside the aGrUM library, the aGrUM framework provides a wrapper for Python: pyAgrum. Included and compiled from the aGrUM repository, the library is also available as python packages: Pypi (https://pypi.org/project/pyagrum/) or Anaconda (https://anaconda.org/conda-forge/pyagrum). The library comes with an online documentation (https://pyagrum.readthedocs.io/) and many tutorials as notebooks (https://agrum.gitlab.io/extra/pyagrum_notebooks.html).

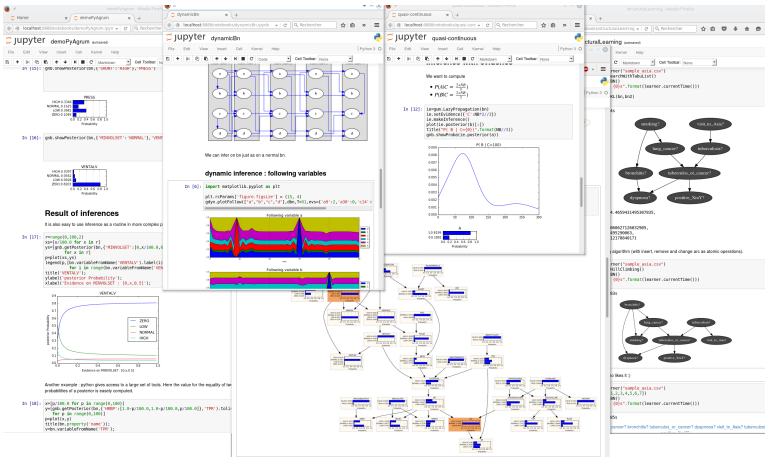

Figure 1: Some Python notebooks using pyAgrum.

pyAgrum is a user friendly high-level python module for manipulating aGrUM's graphical models in python while keeping the high performance level of the C++ library. Within Python Notebooks, pyAgrum can be easily used as a PGM graphical editor. Figure 1 shows such notebooks.

pyAgrum aims to help working with graphical models, not only for model building but also for using these very models in more complex processes and applications. For instance, Figure 2 is taken from one of the many examples provided by the pyAgrum notebooks. It shows how to use pyAgrum to iterate over many probabilistic inferences in order to visualize the impact of the value of an evidence over one variable on another one.

The main purpose of pyAgrum is to provide an extensive set of tools for dealing with GMs that can be used either as is for learning and modeling or as a stable foundation for new and sophisticated algorithms. Specific applications have already been built on top of it. Due to space limitations, we only list the main ones and briefly describe them:

- **Causality**: a module to deal with causal Bayesian networks (including latent variables) and to identify causal impacts (full implementation of the do-calculus) and counterfactuals analysis. See https://agrum.gitlab.io/extra/bookofwhy_notebooks.html for more details.
- **Classifier**: a module to fit and predict classifiers with Bayesian networks in the context of scikit-learn. See 42-CompareClassifiersWithSklearn in https://agrum.gitlab.io/extra/pyagrum_notebooks.html for more details.
- **Copula Bayesian Networks**: new representation and learning algorithms for non-parametric continuous Bayesian networks using (Bernstein) copula. This module is the result of a collaboration between pyAgrum and the OPENTURNS library (http://www.openturns.org/).

Using inference as a function

It is also easy to use inference as a routine in more complex procedure.

In [13]:
```
1  import time
2
3  r=range(0,100,2)
4  xs=[x/100.0 for x in r]
5
6  tf=time.time()
7  ys=[gum.getPosterior(bn,{'MINVOLSET':[0,x/100.0,0.5]},'VENTALV').tolist()
8          for x in r]
9  delta=time.time()-tf
10
11 p=plot(xs,ys)
12 legend(p,[bn.variableFromName('VENTALV').label(i)
13           for i in range(bn.variableFromName('VENTALV').domainSize())],loc=7);
14 title('VENTALV (100 inferences in %d ms)'%delta);
15 ylabel('posterior Probability');
16 xlabel('Evidence on MINVOLSET : [0,x,0.5]');
```

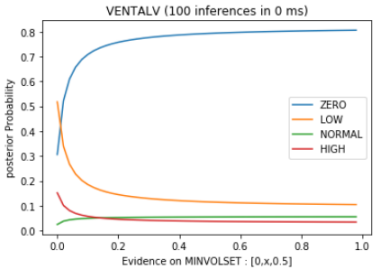

Figure 2: `pyAgrum` in action: sensitivity analysis. The call of an inference on a specific variable of a Bayesian network given a specific evidence is performed 50 times at line 7 with Function `gum.getPosterior`. The x axis represents an increasing belief on the MINVOLSET variable of the classical Bayesian network *Alarm*. The y axis indicates the posterior probabilities of the VENTALV variable given this evidence.

- **Tensor-tree approximation**: new inference algorithms using compact approximate representations of CPTs as low rank tensors (paper presented in PGM'20).

4. Conclusion

This paper has presented `aGrUM/pyAgrum`, a flexible, well maintained and highly efficient framework for manipulating graphical models. It is divided into a C++ library (`aGrUM`) designed to perform computational intensive tasks and a set of user-friendly wrappers, notably `pyAgrum`, that enable to exploit `aGrUM` within high level and easy-to-use programming languages like Python. The `aGrUM`/`pyAgrum` framework is now quite popular since conda reports more than 250K downloads and pyPi reports more than 23K downloads the last six months (see `https://agrum.gitlab.io/pages/downloads-of-pyagrum-packages.html`).

`aGrUM/pyAgrum` is used by academics and industrials around the world, both end-users and algorithm designers. For instance, `aGrUM`'s `O3PRM`s are exploited in ongoing projects with EDF (the French national electricity provider) on risk management and with IBM on the exploitation of probabilities in rule-based expert systems. The BN learning module is exploited in projects with IRSN, the French Institute for Nuclear Safety, as well as with Engie, a world leader in the energy field. Other projects with Airbus Research and the `OpenTURNS` consortium use `aGrUM` for structural learning in non-parametric Copula Bayesian Networks.