Darwinian Networks Library

Jhonatan S. Oliveira and André E. dos Santos

Student Numbers:
200334125 and 200334126

Abstract. *Darwinian networks* (DNs) is a simplification of working with *Bayesian networks* (BNs). We propose a programming library for Darwinian networks (DNs). The library covers adaptation and evolution in DNs, for modelling a problem domain and inference on the model, respectively. The implementation uses C++, expanding over a well established BN library called aGrUM. Furthermore, a wrapper in Python is created over the C++ library, in order to facilitate the development of DN systems with a scripting language.

Description

Darwinian networks (DNs) [1] were proposed as a simplification of working with Bayesian networks (BNs). A BN is a set of *conditional probability tables* (CPTs) and a *direct acyclic graph* (DAG) [2]. The DAG is used to represent graphically the relations between vertices and each vertex is associate with a CPT that quantifies those reslations. On the other hand, a DN directly draws the CPTs as a dashed circle called population. A population has white and black nodes, called combative and docile traits, respectively. Combative traits are used to represent variables on the LHS of a CPT and docile traits represent the variables of the RHS of a CPT. Therefore, a DN is defined as a multi-set of populations.

Modeling in BNs involves testing independencies in a DAG using a graphical test called *d-separation* [2]. Testing independencies in DNs can be graphically performed with a test called *adaptation* [1]. Inference in BN is done by systematically manipulating the CPTs with multiplications and divisions and marginalization. DNs graphically represents multiplications and divisions with the merge of populations and marginalization with replication of populations followed by natural selection. DNs perform both modeling and inference in the same platform.

A programming library is essential to run experimental results, develop real world systems, incentive the literature in the area, among other applications. In this paper, we propose the implementation of a DN library, which contains the basic tools for modeling, testing adaptation and performing evolution in DNs. A user of the library can work with DNs following general ideas from published papers in DNs [1] and discover new practical understanding of BNs, all applied with simple procedures that act with the same remarkable robust view of DNs.

For modeling DNs, the library makes available procedures for all steps of adaptation, first defining populations with combative and docile traits. Then,

to test adaptation between two sets of populations $\mathcal{P}_X$ and $\mathcal{P}_Z$ given a third set of populations $\mathcal{P}_Y$, namely $A(\mathcal{P}_X, \mathcal{P}_Y, \mathcal{P}_Z)$, it is simply necessary to define an initial and a final DN. Moreover, the process of natural selection removes barren populations and the procedure for docilization acts on the resulting DN adding docile populations (populations containing only docile traits) for each population with more than one docile trait. Deletion procedure then removes all populations in $\mathcal{P}_Y$. Finally, a procedure for recursive merging is called in order to check if traits from $\mathcal{P}_X$ and $\mathcal{P}_Z$ end up together in a same population.

For inference, three basic procedures are available, namely, merge, replication and natural selection. First, an initial and final DNs are determined. The merge of two populations combines them into a new population with all combative traits and some of the docile traits. When two combative traits for the same variable occurs that means a division. The library is capable to detect all the possible combination of traits and perform the respective operation. Replication of a population is a procedure which makes a copy of that population in the current DN as well as generated any other population with all the docile from the original one but a subset of the combative traits. Then, natural selection can be called eventually in order to remove unnecessary populations for that evolution up to when the call occurs.

The implementation of the DN library is in C++ programming language. A well established open source C++ framework for BN, called *aGrUM*[1], is used as base for the DN library, meaning that all the basic data structure, like CPTs, potentials, variables, multiplication, addition, and division, among others, is used from aGrUM. Besides of the fast implementation of aGrUM, another advantage of using this framework is to provide a platform for fair comparison between BNs and DNs, since both are in the same programming language and the same set of basic implementation. One salient feature of aGrUm it is a wrapper for Python which uses another library called *Simplified Wrapper and Interface Generator*[2] (SWIG). The DN library will also use SWIG in order to provide a Python programming interface for the users.

In summary, this paper proposes the implementation of a DN library. All the basic operations for adaptation and evolution in DNs will be available through the library. The implementation itself uses C++ programming language and the basic data structures are provided by aGrUM library. A Python interface will be also available for rapid prototyping systems and teaching. The main advantage of the DN library is to apply novel ideas and techniques of DNs that were recently propose to simplify working with BNs.

[1] http://agrum.lip6.fr
[2] http://www.swig.org/

References

1. Butz, C.J., Oliveira, J.S., dos Santos, A.E.: Darwinian networks. In: Twenty-Eighth Canadian Conference on Artificial Intelligence. vol. 9091, pp. 16–29 (2015)
2. Pearl, J.: Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufmann (1988)