Journal of Machine Learning Research 11 (2010) 2169-2173 Submitted 2/10; Revised 8/10; Published 8/10

libDAI: A Free and Open Source C++ Library for Discrete Approximate Inference in Graphical Models

Joris M. Mooij JORIS.MOOIJ@LIBDAI.ORG
Max Planck Institute for Biological Cybernetics
Department for Empirical Inference
Spemannstraße 38, 72076 Tübingen
Germany

Editor: Cheng Soon Ong

Abstract

This paper describes the software package libDAI, a free & open source C++ library that provides implementations of various exact and approximate inference methods for graphical models with discrete-valued variables. libDAI supports directed graphical models (Bayesian networks) as well as undirected ones (Markov random fields and factor graphs). It offers various approximations of the partition sum, marginal probability distributions and maximum probability states. Parameter learning is also supported. A feature comparison with other open source software packages for approximate inference is given. libDAI is licensed under the GPL v2+ license and is available at `http://www.libdai.org`.

Keywords: probabilistic graphical models, approximate inference, open source software, factor graphs, Markov random fields, Bayesian networks

1. Introduction

Probabilistic graphical models (Koller and Friedman, 2009) are at the core of many applications in machine learning nowadays. Because exact inference in graphical models is often infeasible, an extensive literature has evolved in recent years describing various approximation methods for this task. Unfortunately, there exist only few free software implementations for many of these methods. The software package libDAI presented in this paper is the result of an ongoing attempt to improve this situation.

libDAI is a free and open source C++ library (licensed under the GNU General Public License version 2, or higher) that provides implementations of various exact and approximate inference methods for graphical models. libDAI supports factor graphs with discrete variables. This paper describes the most recent libDAI release, version 0.2.7.

The modular design of libDAI makes it easy to implement novel inference algorithms. The large number of inference algorithms already implemented in libDAI allows for easy comparison of accuracy and performance of various algorithms.

2. Inference in Factor Graphs

Although Bayesian networks and Markov random fields are the most commonly used graphical models, libDAI uses a slightly more general type of graphical model: factor graphs (Kschischang

et al., 2001). A *factor graph* is a bipartite graph consisting of *variable* nodes $i \in \mathcal{V}$ and *factor* nodes $I \in \mathcal{F}$, a family of random variables $(X_i)_{i\in\mathcal{V}}$, a family of *factors* $(f_I)_{I\in\mathcal{F}}$ (which are nonnegative functions depending on subsets $N_I \subseteq \mathcal{V}$ of the variables), and a set of edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{F}$, where variable node i and factor node I are connected by an undirected edge if and only if the factor f_I depends on variable X_i, that is, if $i \in N_I$. The probability distribution encoded by the factor graph is given by:

$$\mathbb{P}(X_{\mathcal{V}}) = \frac{1}{Z}\prod_{I\in\mathcal{F}} f_I(X_{N_I}), \qquad Z := \sum_{X_{\mathcal{V}}}\prod_{I\in\mathcal{F}} f_I(X_{N_I}).$$

A Bayesian network can easily be represented as a factor graph by introducing a factor for each variable, namely the (conditional) probability table of the variable given its parents in the Bayesian network. A Markov random field can be represented as a factor graph by taking the clique potentials as factors. Factor graphs naturally express the factorization structure of probability distributions. Therefore, they form a convenient representation for approximate inference algorithms that exploit this factorization. This is the reason that libDAI uses factor graphs as the basic representation for graphical models.

Given a factor graph, the following important inference tasks can be distinguished: calculating the *partition sum* Z; calculating *marginal* probability distributions $\mathbb{P}(X_A) = \frac{1}{Z}\sum_{X_{\mathcal{V}\setminus A}}\prod_{I\in\mathcal{F}} f_I(X_I)$ over subsets of variables X_A, $A \subseteq \mathcal{V}$; calculating the *Maximum A Posteriori* (MAP) state, that is, the joint configuration $\operatorname{argmax}_{X_{\mathcal{V}}}\prod_{I\in\mathcal{F}} f_I(X_I)$ which has the maximum probability mass. libDAI offers several inference algorithms which solve these tasks either approximately or exactly, for factor graphs with discrete variables.

2.1 Inference Methods Implemented in libDAI

Apart from exact inference by brute force enumeration and the junction-tree method, libDAI currently offers the following approximate inference methods for calculating partition sums, marginals and MAP states:[1] mean field, (loopy) belief propagation (Kschischang et al., 2001), fractional belief propagation (Wiegerinck and Heskes, 2003), tree-reweighted belief propagation (Wainwright et al., 2003), tree expectation propagation (Minka and Qi, 2004), generalized belief propagation (Yedidia et al., 2005), double-loop generalized belief propagation (Heskes et al., 2003), loop-corrected belief propagation (Mooij and Kappen, 2007; Montanari and Rizzo, 2005), conditioned belief propagation (Eaton and Ghahramani, 2009), a Gibbs sampler and a decimation method.

In addition, libDAI supports parameter learning of conditional probability tables by maximum likelihood or expectation maximization (in case of missing data).

3. Technical Details

libDAI is targeted at researchers that have a good understanding of graphical models. The best way to use libDAI is by writing C++ code that directly invokes the library functions and classes. In addition, part of the functionality is accessible by using various interfaces: a command line interface, a limited MatLab interface, and experimental Python and Octave interfaces (powered by SWIG, see `http://www.swig.org`). Because libDAI is implemented in C++, it is very fast compared with implementations in pure MatLab or Python; the difference in computation time can be up to a few orders of magnitude.

1. Not all inference tasks are implemented by each method.

	libDAI	BNT	PNL	FastInf	GRMM	Factorie	JProGraM
Language	C++	MatLab, C	C++	C++	Java	Scala	Java
License	GPL2+	LGPL2	BSD	GPL3	CPL 1.0	CC-by 3.0	GPL3
Junction tree	+	+	+	+	+	-	+
Belief propagation (BP)	+	+	+	+	+	+	-
Fractional BP	+	-	-	-	-	-	-
Tree-reweighted BP	+	-	-	+	-	-	-
Generalized BP (GBP)	+	-	-	+	+	-	-
Double-loop GBP	+	-	-	-	-	-	-
Loop-corrected BP	+	-	-	-	-	-	-
Conditioned BP	+	-	-	-	-	-	-
Tree expectation propagation	+	-	-	-	-	-	-
Mean field	+	-	-	+	-	-	-
Gibbs sampling	+	+	+	+	+	+	+
Other sampling methods	-	+	+	+	+	+	-
Decimation	+	-	-	-	-	-	-
Continuous variables	-	+	+	-	-	+	+
Dynamic Bayes nets	-	+	+	-	-	+	-
Conditional random fields	-	-	-	-	+	+	-
Relational models	-	-	-	+	-	+	-
Influence diagrams	-	+	-	-	-	-	-
Parameter learning	+	+	+	+	+	+	+
Structure learning	-	+	+	-	-	-	+

Table 1: Feature comparison of various open source software packages for approximate inference on graphical models.

libDAI is designed to be cross-platform and is known to compile on GNU/Linux distributions and on Mac OS X using the `GCC` compiler suite, and also under Windows using either `CygWin` or `MS Visual Studio 2008`.

The libDAI sources and documentation can be downloaded or browsed from the libDAI website at `http://www.libdai.org`. The Google group libDAI (see `http://groups.google.com/group/libdai`) can be used for getting support and discussing development issues. Extensive documentation generated by Doxygen (see `http://www.doxygen.org`) is available, both as part of the source code and online. Unit tests and various example programs are provided, including the full source code of the solver which won in two categories of the *2010 UAI Approximate Inference Challenge* (see also `http://www.cs.huji.ac.il/project/UAI10/`) and uses several of the algorithms implemented in libDAI.

4. Related Work

Other prominent open source software packages supporting both directed and undirected graphical models are the Bayes Net Toolbox (BNT, see `http://bnt.googlecode.com`), the Probabilistic Networks Library (PNL, see `http://sourceforge.net/projects/openpnl`), FastInf (Jaimovich et al., 2010), GRMM (`http://mallet.cs.umass.edu/grmm`), Factorie (`http://code.google.com/p/factorie`), and JProGraM (`http://jprogram.sourceforge.net`). A feature comparison of libDAI with these other packages is provided in Table 1.

Acknowledgments

Part of this work was part of the Interactive Collaborative Information Systems (ICIS) project, supported by the Dutch Ministry of Economic Affairs, grant BSIK03024. I would like to express my gratitude to the following people who made major contributions to libDAI: Martijn Leisink (for laying down the foundations for the library), Frederik Eaton (for contributing the Gibbs sampler, conditioned BP, fractional BP and various other improvements), Charles Vaske (for contributing the parameter learning algorithms), Giuseppe Passino (for various improvements), Bastian Wemmenhove (for contributing the MR algorithm), Patrick Pletscher (for contributing the SWIG interface). I am also indebted to the following people for bug fixes and miscellaneous smaller patches: Claudio Lima, Christian Wojek, Sebastian Nowozin, Stefano Pellegrini, Ofer Meshi, Dan Preston, Peter Gober, Jiuxiang Hu, Peter Rockett, Dhruv Batra, Alexander Schwing, Alejandro Lage, Matt Dunham.

References

F. Eaton and Z. Ghahramani. Choosing a variable to clamp. In *Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics (AISTATS 2009)*, volume 5 of *JMLR Workshop and Conference Proceedings*, pages 145–152, 2009.

T. Heskes, K. Albers, and B. Kappen. Approximate inference and constrained optimization. In C. Meek and U. Kjærulff, editors, *Proceedings of the Nineteenth Annual Conference on Uncertainty in Artificial Intelligence (UAI-03)*, pages 313–320. Morgan Kaufmann, 2003.

A. Jaimovich, O. Meshi, I. McGraw, and G. Elidan. FastInf: An efficient approximate inference library. *Journal of Machine Learning Research*, 11:17331736, 2010.

D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.

F. Kschischang, B. Frey, and H.-A. Loeliger. Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, 47(2):498–519, 2001.

T. Minka and Y. Qi. Tree-structured approximations by expectation propagation. In S. Thrun, L. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems 16*, pages 193–200. MIT Press, 2004.

A. Montanari and T. Rizzo. How to compute loop corrections to the Bethe approximation. *Journal of Statistical Mechanics: Theory and Experiment*, 2005(10):P10011, 2005.

J. Mooij and B. Kappen. Loop corrections for approximate inference on factor graphs. *Journal of Machine Learning Research*, 8:1113–1143, 2007.

M. Wainwright, T. Jaakkola, and A. Willsky. Tree-reweighted belief propagation algorithms and approximate ML estimation by pseudo-moment matching. In C. Bishop and B. Frey, editors, *Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics (AISTATS 2003)*, 2003.

W. Wiegerinck and T. Heskes. Fractional belief propagation. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 438–445. MIT Press, 2003.

J. Yedidia, W. Freeman, and Y. Weiss. Constructing free-energy approximations and generalized belief propagation algorithms. *IEEE Transactions on Information Theory*, 51(7):2282–2312, 2005.