AMIDST: a Java Toolbox for Scalable Probabilistic Machine Learning

Andrés R. Masegosa[a,*], Ana M. Martínez[b,*], Darío Ramos-López[a,*], Rafael Cabañas[a,*], Antonio Salmerón[a], Helge Langseth[c], Thomas D. Nielsen[b], Anders L. Madsen[d,b]

[a] *University of Almería, ES-04120 Almería, Spain*
[b] *Aalborg University, DK-9220 Aalborg, Denmark*
[c] *Norwegian University of Science and Technology, NO-7491 Trondheim, Norway*
[d] *HUGIN EXPERT A/S, DK-9000 Aalborg, Denmark*

Abstract

The AMIDST Toolbox is an open source Java software for scalable probabilistic machine learning with a special focus on (massive) streaming data. The toolbox supports a flexible modelling language based on probabilistic graphical models with latent variables. AMIDST provides parallel and distributed implementations of scalable algorithms for doing probabilistic inference and Bayesian parameter learning in the specified models. These algorithms are based on a flexible variational message passing scheme, which supports discrete and continuous variables from a wide range of probability distributions.

Keywords: Probabilistic Graphical Models, Scalable algorithms, Variational methods, Latent variables

1. Introduction

AMIDST[1] is a toolbox for the analysis of large-scale data sets using probabilistic graphical models (PGMs). These are the so-called openbox models in the sense that PGMs can be easily interpreted by the users. PGMs consist of two parts: a qualitative component in the form of a graph encoding conditional independencies, and a quantitative component consisting of a collection of local probability distributions adhering to the independence properties specified in the graph. Collectively, the two components provide a compact representation of the joint probability distribution over the set of variables in the domain being modelled.

AMIDST implements parallel and distributed algorithms for Bayesian inference and learning in PGMs with latent (or unobserved) variables. The key point of this software is the use of variational methods [6] for making approximate inference. This makes

*These four authors are considered as first authors and contributed equally to this work.

Email addresses: andresmasegosa@ual.es (Andrés R. Masegosa), ana@cs.aau.dk (Ana M. Martínez), dramoslopez@ual.es (Darío Ramos-López), rcabanas@ual.es (Rafael Cabañas), antonio.salmeron@ual.es (Antonio Salmerón), helgel@idi.ntnu.no (Helge Langseth), tdn@cs.aau.dk (Thomas D. Nielsen), anders@hugin.dk (Anders L. Madsen)

[1]For brevity, we will refer to the AMIDST Toolbox as either AMIDST or the toolbox.

AMIDST suitable for analysing streaming data because our models can efficiently be updated when new data is available. Numerous tools for graphical models have been published during the last three decades[2]. However, the vast majority of them do not support scalable inference and learning algorithms. To the best of our knowledge, there is no existing software for mining data streams based on PGMs (including latent variable models); most existing tools focus on stationary data sets [8]. Additionally, the implemented methods can be launched in computing clusters running either Apache Flink or Apache Spark.

2. Background

2.1. Probabilistic graphical models

AMIDST supports the specification of *Bayesian networks (BNs)* [9, 2], which are widely used PGMs for reasoning under uncertainty. Formally, let $\boldsymbol{X} = \{X_1, \ldots, X_N\}$ denote the set of random variables defining our problem domain. BNs can be represented by a directed acyclic graph (DAG). Each node, labelled X_i, is associated with a factor or conditional probability $p(X_i|\, pa(X_i))$, where $pa(X_i) \subset \boldsymbol{X} \setminus X_i$ represents the so-called *parent variables* of X_i, i.e., the variables corresponding to the parent nodes of X_i in the graph. A BN defines a joint distribution $p(\boldsymbol{X})$ in the following form:

$$p(\boldsymbol{X}) = \prod_{i=1}^{N} p(X_i|\, pa(X_i)). \tag{1}$$

For modelling problems where variables have continuous state spaces, the AMIDST Toolbox allows the specification of conditional linear Gaussian (CLG) Networks [4, 5]. Furthermore, latent (i.e., hidden) variables are supported. These variables cannot be observed and are included in the model to capture correlation structure. The use of latent variables allows the representation of a large range of problems with complex probabilistic dependencies.

2.2. Scalable inference with variational methods

Inference (a.k.a. belief updating) in PGMs typically corresponds to calculating the posterior distribution $p(\boldsymbol{X}_I = \boldsymbol{x}_I|\boldsymbol{X}_E = \boldsymbol{x}_E)$, where $\boldsymbol{X}_E \subset \boldsymbol{X}$ is the set of observed variables and $\boldsymbol{X}_I$ is the set of variables of interest with $\boldsymbol{X}_I \subseteq \boldsymbol{X} \setminus \boldsymbol{X}_E$.

Variational inference is a deterministic approximate inference technique, where we seek to iteratively optimise a variational approximation to the posterior distribution of interest [1]. Let $\mathcal{Q}$ be the set of possible approximations; then the variational approximation to a posterior distribution $p(\boldsymbol{x}_I|\boldsymbol{X}_E = \boldsymbol{x}_E)$ is defined as

$$q^*_{\boldsymbol{x}_E}(\boldsymbol{x}_I) = \arg\min_{q \in \mathcal{Q}} D(q(\boldsymbol{x}_I)||p(\boldsymbol{x}_I|\boldsymbol{X}_E = \boldsymbol{x}_E)),$$

where $D(q||p)$ is the Kullback-Leibler divergence between q and p. In the AMIDST Toolbox, the variational inference scheme employs a so-called mean-field approximation, which roughly assumes that the variables of interest are pairwise independent

[2]See this link for an updated list `http://www.cs.ubc.ca/~murphyk/Software/bnsoft.html`.

given the observed evidence; in turn this means that the posterior variational distribution factorises over the variables involved, i.e., $q^*_{\boldsymbol{x}_E}(\boldsymbol{x}_I) = \prod_{i \in I} q^*_{\boldsymbol{x}_E}(x_i)$. Optimising the variational approximation can be achieved by using either coordinate or gradient ascent (or a stochastic approximation thereof), which guarantees convergence towards a (local) optimum.

Learning the parameters $\boldsymbol{\theta}$ of a BN from a training data set D can be reduced to the inference task of computing $p(\boldsymbol{\theta}|D)$. With this consideration, the AMIDST Toolbox implements a general architecture for supporting the *variational message passing* (VMP) algorithm [11] in PGMs. This algorithm can be easily scaled-up as messages are independent. In particular, two versions are provided; a parallel version exploiting multi-core architectures, powered by Java 8 Streams [7]; and a novel distributed version, named d-VMP [6], for large-scale data processing on computing clusters running either Apache Flink or Apache Spark.

3. Software framework

3.1. Functionalities

The key functionalities of the toolbox are summarised as follows:

- **Openbox models**: with the specification of PGMs, AMIDST's approach to machine learning is based on the use of openbox models that can be inspected and which can incorporate prior information or knowledge about the domain, in contrast to other approaches which cannot be interpreted by the users.

- **Efficient belief updating**: this toolbox implements, among others, approximate Bayesian inference algorithms based on variational methods (see Section 2.2). This allows for an efficient updating of the models which is suitable in cases where the whole data cannot be stored in memory.

- **Multi-core and distributed learning**: AMIDST provides parallel and distributed implementations of variational algorithms [11] that can be run on multi-core CPUs, using Java 8's built-in functionalities, or in massive data sets by interfacing with Apache Flink and Apache Spark. Further details and experimental results about these methods can be found in [6, 7].

3.2. Architecture

AMIDST has been designed following a modular structure. This allows future extensions to be made independently of the core design, thereby leaving the kernel small and robust. Another added value of the modularity is that it enables a more seamless interaction with external software. Currently, AMIDST interfaces with MOA, Weka, and HUGIN [3]. The toolbox is distributed using Maven[4]. The use of this technology simplifies the installation making the interaction with external software transparent.

[3] MOA: `http://moa.cms.waikato.ac.nz`, Weka: `http://www.cs.waikato.ac.nz/ml/weka/`, and HUGIN: `http://www.hugin.com`.

[4] `https://maven.apache.org`

4. Illustrative examples

In this section we illustrate the use of AMIDST in multi-core and parallel architectures[5]. In particular, we consider the classification model proposed in [3] and a dataset used in genetics [10] (which contains about 500,000 instances and which has been split into files of 100,000 instances).

The `DataStream` class in package `eu.amidst.core.datastream` is an interface for dealing with data streams in a single computer. The toolbox is designed to process the data sequentially without having to load all observations into main memory simultaneously. The functionality for loading data is provided by class `DataStreamLoader`. The following code provides an example of reading data from a *.arff* file (Weka file format):

```
DataStream data = DataStreamLoader.open("codrnaNorm_100k_1.arff");
```

When we have a massive data set which does not fit into a single computer, we can use a Big Data framework like Apache Flink to deal with data sets stored in a distributed computing cluster. For reading these data sets we can use the class `eu.amidst.flink.data.DataFlink`, as shown in the next code fragment:

```
//Set-up Flink Session
ExecutionEnvironment env = ExecutionEnvironment.getExecutionEnvironment();
// Load the distributed data
DataFlink<DataInstance> data =
            DataFlinkLoader.open(env, "hdfs://codrnaNorm_100k_1.arff", false);
```

AMIDST contains a wide range of predefined models, most of them including latent variables (and custom models can also be defined by the user). These models are available in the *latent-variable-models* module. Learning is straight-forward as shown in the next code fragment, which also illustrates the toolbox's seamless handling of massive data sets during model learning/updating; the code is valid for both `DataStream` and `DataFlink` objects. Lines 4 to 8 show how the model can be updated in case new data sets become available.

```
1  Model model = new LatentClassificationModel(data.getAttributes())
2                      .setClassName("codrna_Y")
3
4  for(int i=1; i<=5; i++) {
5      if (i > 1) data = DataStreamLoader.open("codrnaNorm_100k_"+i+".arff");
6      model.updateModel(data);
7      System.out.println(model.getModel());
8  }
```

[5] Visit `https://github.com/amidst/example-project` for downloading an easy to run project with these examples.

AMIDST's webpage (`www.amidsttoolbox.com`) contains a large class of code examples covering all the functionalities of the toolbox.

Acknowledgments

This work was performed as part of the AMIDST project. AMIDST has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement no 619209. AM, DRL and AS thank the support from CDTIME. DRL thanks also to CEIMAR.

References

[1] H. Attias. A variational Bayesian framework for graphical models. *Advances in neural information processing systems*, pages 209—215, 2000.

[2] F.V. Jensen and T.D. Nielsen. *Bayesian Networks and Decision Graphs*. Springer Publishing Company, Incorporated, second edition, 2007.

[3] H. Langseth and T.D. Nielsen. Latent classification models. *Machine Learning*, 59(3):237–265, 2005.

[4] S.L. Lauritzen. Propagation of probabilities, means, and variances in mixed graphical association models. *Journal of the American Statistical Association*, 87(420):1098–1108, 1992.

[5] S.L. Lauritzen. *Graphical Models*. Oxford University Press, 1996.

[6] A.R. Masegosa, A. M. Martínez, H. Langseth, T.D. Nielsen, A. Salmerón, D. Ramos-López, and A.L. Madsen. Scaling up Bayesian variational inference using distributed computing clusters. *International Journal of Approximate Reasoning*, 88:435–451, 2017.

[7] A.R. Masegosa, A.M Martínez, and H. Borchani. Probabilistic graphical models on multi-core CPUs using Java 8. *IEEE Computational Intelligence Magazine*, 11(2):41–54, 2016.

[8] K.P. Murphy. Software for graphical models: A review. *International Society for Bayesian Analysis Bulletin*, 14(4):13–15, 2007.

[9] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers, San Mateo, CA., 1988.

[10] A.V. Uzilov, J.M. Keegan, and D.H. Mathews. Detection of non-coding RNAs on the basis of predicted secondary structure formation free energy change. *BMC bioinformatics*, 7(1):173, 2006.

[11] J.M. Winn and C.M. Bishop. Variational message passing. *Journal of Machine Learning Research*, 6:661–694, 2005.

Required Metadata

Current executable software version

Nr.	(executable) Software metadata description	Please fill in this column
S1	Current software version	0.7.1
S2	Permanent link to executables of this version	https://github.com/amidst/toolbox/releases/tag/v0.7.1
S3	Legal Software License	Apache 2.0
S4	Computing platform/Operating System	Linux, OS X, Microsoft Windows
S5	Installation requirements & dependencies	Maven, Java 8
S6	If available, link to user manual - if formally published include a reference to the publication in the reference list	http://www.amidsttoolbox.com/documentation/
S7	Support email for questions	contact@amidsttoolbox.com

Table 1: Software metadata

Current code version

Nr.	Code metadata description	Please fill in this column
C1	Current code version	0.7.1
C2	Permanent link to code/repository used of this code version	https://github.com/amidst/toolbox
C3	Legal Code License	Apache 2.0
C4	Code versioning system used	git
C5	Software code languages, tools, and services used	Java 8
C6	Compilation requirements, operating environments & dependencies	Maven
C7	If available Link to developer documentation/manual	http://www.amidsttoolbox.com/documentation/
C8	Support email for questions	contact@amidsttoolbox.com

Table 2: Code metadata