Contents

Document history

Version	Date	Author (Unit)	Description
v0.3	21/11 2014	Helge Langseth, Thomas D. Nielsen, Antonio Salmerón	First draft

1 Executive summary

The aim of this document is to describe the progress of the software development related to learning in the AMIDST project at the end of the first year.

2 Introduction

[[This is just the first page from D3.1, then quickly terminated after learning is introduced. Just to say what a BN is.]]

Probabilistic graphical models provide a well-founded and principled approach for performing inference in complex domains endowed with uncertainty. A probabilistic graphical model is a framework consisting of two parts: a qualitative component in the form of a graphical model encoding conditional independence assertions about the domain being modelled as well as a quantitative component consisting of a collection of local probability distributions adhering to the independence properties specified in the graphical model. Collectively, the two components provide a compact representation of the joint probability distribution over the domain being modelled.

Bayesian networks (BNs) [**?**] are a particular type of probabilistic graphical model that has enjoyed widespread attention in the last two decades. Figure 1 shows a BN representing the joint distribution of variables $X_1, \ldots, X_5$. Attached to each node, there is a conditional probability distribution given its parents in the network, so that the joint distribution factorises as

$$p(X_1, \ldots, X_5) = p(X_1)p(X_2|X_1)p(X_3|X_1)p(X_4|X_2, X_3)p(X_5|X_3).$$

In general, for a BN with n variables $\mathbf{X} = \{X_1, \ldots, X_n\}$, the joint distribution factorises as

$$p(\mathbf{X}) = \prod_{i=1}^{n} p(X_i|\mathrm{pa}\,(X_i)), \tag{1}$$

where $\mathrm{pa}\,(X_i)$ denotes the set of parents of X_i in the network.

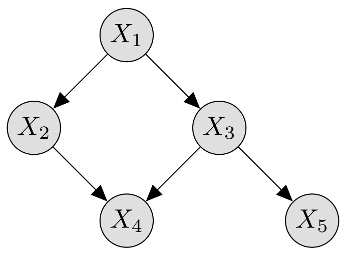


Figure 1: A Bayesian network with five variables.

We will use lowercase letters to refer to values or configurations of values, so that x denotes a value of X and $\mathbf{x}$ is a configuration of the variables in $\mathbf{X}$. Given a set of observed variables $\mathbf{X}_E \subset \mathbf{X}$ and a set of variables of interest $\mathbf{X}_I \subset \mathbf{X} \setminus \mathbf{X}_E$, *probabilistic* *inference* is the calculation of the posterior distribution $p(x_i|\mathbf{x}_E)$ for each $i \in I$. A thorough introduction to the state of the art for inference in Bayesian networks was given in [**?**].

In this document we assume that inference techniques for a given Bayesian network is available, and will consider how to define a Bayesian network model that fits a data set as well as possible. This process is known as *learning* in the Bayesian network community.

3 Learning Bayesian networks

Consider again the factorization of the full joint distribution $p(\mathbf{x})$ in Equation (1). With this factorization we can efficiently represent the joint probability distribution $p(\mathbf{x})$, but it requires that $\mathrm{pa}\,(X_i)$, i.e., the *parent set* of X_i, is known for each $i = 1, \ldots, n$. The parents of X_i are exactly the nodes which have an outgoing edge pointing to X_i (e.g., $\mathrm{pa}\,(X_4) = \{X_2, X_3\}$ for the model in Figure 1). Algorithms for learning the parent sets, also known as *structural learning* algorithms, follow one of two approaches: *i*) search and score methods, see e.g. [**?**] and *ii*) constraint-based techniques [**?**]. Both will be considered in Task 4.1 of the project, and the progress towards the implementation of structural learning in AMIDST is summarized in Section 5.

As soon as the parent sets in Equation (1) are determined, the specific conditional distributions $p(X_i|\mathrm{pa}\,(X_i))$ can also be learned from data. The approach taken in the AMIDST project is that the conditional distribution $p(X_i|\mathrm{pa}\,(X_i))$ is assumed to be a member of the exponential family (including, among others, the Gaussian and multinomial distributions). Therefore, parameter learning amounts to finding the optimal *parameterization* of the given distributions. This task will be considered in Task 4.2 and Task 4.4.

Let $\boldsymbol{\theta}$ denote the combination of all parameters of the model (that is, all parameterizations of the local conditional distribution functions in Equation (1)), and let $\mathcal{D} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ denote the dataset which we are learning from. $\mathcal{D}$ is a collection of N independent configurations over the variables of the domain. Note that $\mathcal{D}$ may be *incomplete*, in case some of the observations are missing. Missingness can occur, for instance, if some of the variables are latent, or some data-collecting instrument fails.

Now, parameter learning in Bayesian networks amounts to estimating $\boldsymbol{\theta}$ using some optimization criterion that depend on $\mathcal{D}$. The learning generally comes in two different shapes. The most commonly used approach (at least traditionally) is *maximum likelihood* learning, which attempts to find the model parameters that maximizes the *likelihood* of the model given the data. The likelihood of the parameters given the data set $\mathcal{D}$, is defined as $\mathcal{L}(\boldsymbol{\theta}|\mathcal{D}) = \prod_{j=1}^{N} p(\mathbf{x}_j|\boldsymbol{\theta})$, and the maximum likelihood estimator is $\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}|\mathcal{D})$.

Maximum likelihood learning is very efficient in exponential family models when there are no missing observations in $\mathcal{D}$, because the parameters of each distribution can be optimized independently. However, closed-form solutions are not available when the dataset contains missing values.

Missing values is, though, a key ingredient in the data sets entertained in the AMIDST project. In these cases, maximum likelihood learning typically relies on the Expectation Maximization (EM) algorithm or generalizations thereof. Implementations of the (generalized) EM algorithm iterates over the following two steps that are repeated until convergence: *i*) E-step: Inference in the model given the data-set (using a model with the current parameterization); *ii*) M-step: Updates of the parameters using the posterior distributions calculated in the E-step. The EM algorithm is a greedy algorithm, that ensures that the likelihood of the current parameter estimates is non decreasing from one iteration to the next, and therefore also converges to a (local) maximum of the likelihood function. Unfortunately, this is only guaranteed when an exact E-step is conducted, and if one uses approximative inference the algorithm may not improve as it moves along. Exact inference in the E-step is computational prohibitive in a streaming context; additionally, the M-step can also be computationally challenging. Maximum likelihood learning is therefore not implemented in the open-source software of the AMIDST project, but can be employed utilizing the interface that the toolbox offers to the implementation inside the Hugin system.

Alternatively to the maximum likelihood approach we have the *Bayesian* paradigm. From a simplistic point of view, the main difference between the two is that the Bayesian set-up regards $\boldsymbol{\theta}$ as a vector of *random variables*, and treats them on an equal footing as all other (unobservable) variables in the domain. From a modelling perspective this extension is straight forward: The model in Figure 2 extends the model in Figure 1 by explicitly representing $\boldsymbol{\theta} = \{\theta_i\}_{i=1}^5$, where θ_i is the parameterization of the conditional distribution function $p(X_i|\mathrm{pa}\,(X_i))$. Parameter learning now amounts to calculating the probability distribution $p(\boldsymbol{\theta}|\mathcal{D})$, that is, it is reduced to inference in an (extended) Bayesian network model. Thereby, approximate inference techniques like variational Bayes message passing, exponential propagation or importance sampling can be used directly for learning.

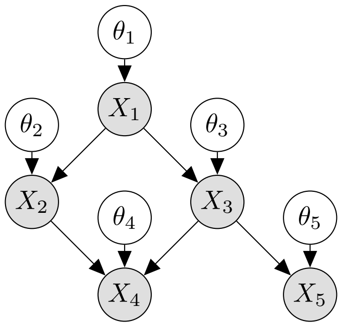


Figure 2: The Bayesian network from Figure 1 extended with explicit representation of the (unknown) parameters.

In practice, the *a priori* marginal distributions for each θ_i must be declared before the learning can be performed. These prior distributions enables domain experts to express knowledge about the parameterization of the distributions that is combined with information from the dataset to obtain the posterior information captured by $p(\boldsymbol{\theta}|\mathcal{D})$. The prior information consists of the definition of a distributional family for each θ_i together with a parameterization (the so-called *hyper-parameters*). In AMIDST we will ensure efficient calculation of posteriors by enforcing the distributional families be the conjugate distribution of the likelihood-terms. The hyper-parameters are chosen freely, and this is the vehicle provided to encode prior (expert) knowledge.

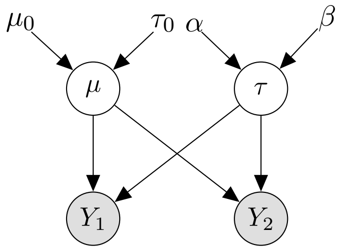


Figure 3: A detailed Bayesian network for the streaming variable Y_t (observed at time $t = 1$ and $t = 2$). Y_t is assumed to follow a Gaussian distribution with mean μ and precision τ.

Figure 3 shows a detailed description of the model for a streaming variable $Y_t \sim N\left(\mu, \tau^{-1}\right)$ observed at $t = 1$ and $t = 2$. Prior information about these parameters are encoded by assuming $\mu \sim N\left(\mu_0, \tau_0^{-1}\right)$ and $\tau \sim \Gamma(\alpha, \beta)$ for given values of the hyper-parameters $\{\mu_0, \tau_0, \alpha, \beta\}$. The dataset $\mathcal{D} = \{y_1, y_2\}$ enables us to calculate

$$p(\mu, \tau|y_1, y_2, \mu_0, \tau_0, \alpha, \beta) = \frac{p(y_1|\mu, \tau)p(y_2|\mu, \tau)p(\mu|\mu_0, \tau_0)p(\tau|\alpha, \beta)}{p(y_1, y_2|\mu_0, \tau_0, \alpha, \beta)}.$$

The calculation is efficient both in terms of both space and time because the model is from the conjugate exponential family.

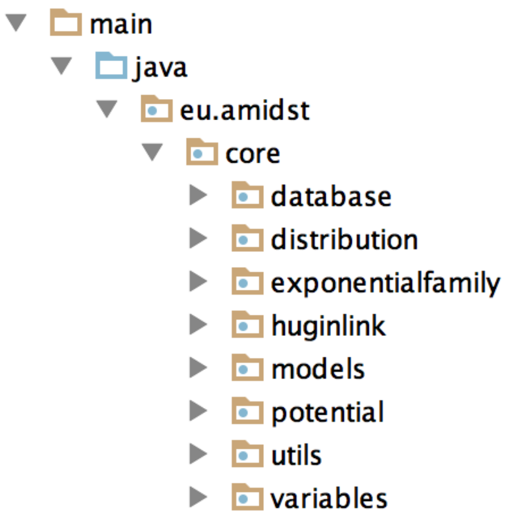
It follows that the learning functionality in AMIDST will rest heavily on the implementation of the core components in the toolbox (variables, distributions, Bayesian networks, and so on) as well the design to accommodate efficient inference using these core components. The implementation of inference engines will constitute a key component for the learning implementation because efficient and scalable *inference* is both a requirement and a guarantee for efficient and scalable *parameter learning*. The next section will therefore discuss the top-level design of the AMIDST toolbox, first describing the core components, thereafter moving to the design of the inference components.

4 TDN – Design

[[This section will cover the design of the SW, as specified through the requirements engineering document. Only the parts related to learning will be covered.]]

As indicated in description of work (DOW) document, the objective of the AMIDST project is to provide an open source toolbox implementing novel developments in scalable algorithms for inference and learning with probabilistic graphical models able to operate in hybrid domains.

The initial developed structure of AMIDST toolbox includes eight packages as shown in Figure 4.



```
main
  java
    eu.amidst
      core
        database
        distribution
        exponentialfamily
        huginlink
        models
        potential
        utils
        variables
```

Figure 4: The main packages defined for Amidst toolbox.

4.1 Design: The learning module

The core components of the modeling framework has been designed and implemented to facilitate an easy integration with the inference and learning modules developed as part of Work packages 3 and 4. Figure 5 shows a high-level overview of the key components of the AMIDST software tool that is directly related to learning; these learning-related components are connected to the framework core (see above) through the Inference component and .... (highlighted using boxes without rounded corners). The color coding in the figure summarizes the implementation status: blue boxes represent software components that have been implemented in the AMIDST toolbox and green boxes represent components that are part of the software design specification but which has not yet been implemented.

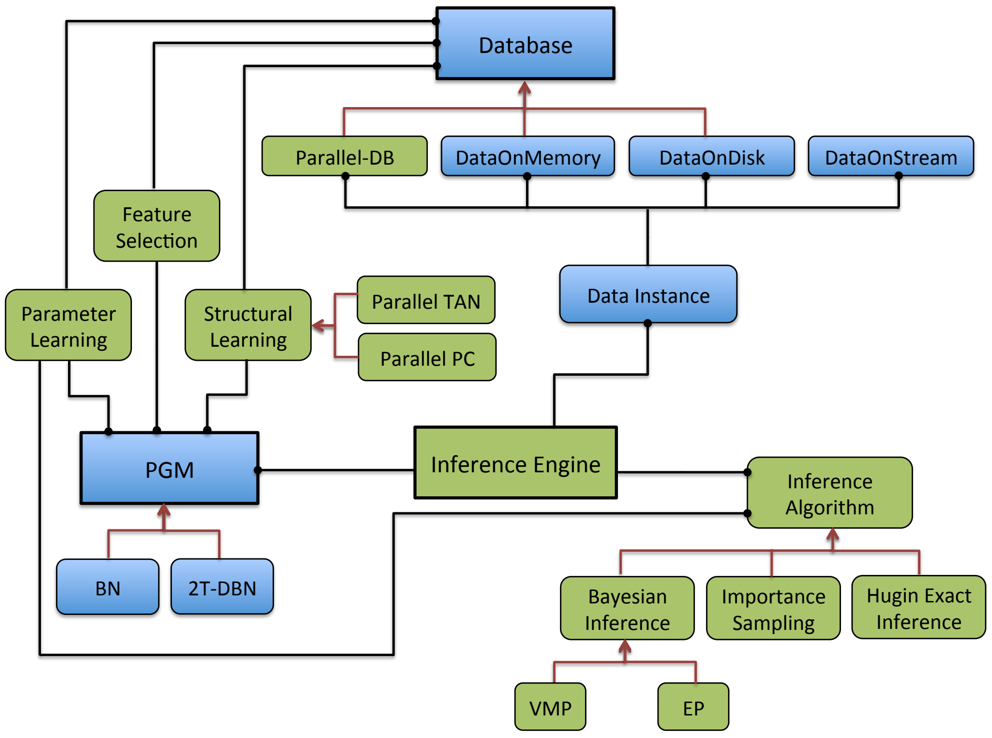


Figure 5: Illustration of the design of the software components related to model learning. Nomenclature: The boxes in the figure represent software components (sets, possibly singletons, of classes), a rounded-arc going from X to Y indicate that Y 'uses/references' X, and an arc with an arrow from X to Y implies inheritance.

As described in Section **??** we pursue a fully Bayesian approach for doing parameter learning in the AMIDST framework. This, in turn, means that parameter learning reduces to the task of inference for which we plan to consider two approaches: variational message passing (positioned in a variational Bayes framework) and expectation propagation. Particular efficient implementations of these methods can be realized when the distribution families of the models are conjugate-exponential. In this case the inference operations can be further supported by specifying the exponential distributions using their natural parameters, cf. Section 3. These improvement are realized through tailored exponential family implementations of the standard distributions that are part of the AMIDST framework (such as conditional linear Gaussian distributions).

In the AMIDST framework we consider two types of data sources: i) Streaming data, where data arrives at high frequency with no storage of historical data (except for data in the most recent past, which is stored in a buffer), and 2) static databases that simply correspond to traditional databases. The database support is realized by a general database component (Database) that defines the database interface from which more specialized databases (DataOnMemory, DataOnDisk, and ParallelDatabase) can be derived:

- DataOnMemory implements database functionality for data sets that can be loaded into main memory.
- DataOnDisk provides functionality for handling datasets too larger to be loaded in main memory.
- ParallelDatabase implements a distributed database.

Note that with this design future users and developers on the AMIDST toolbox can design and implement their database specifications with the only restriction that it should implement the interface defined by the Database component.

Functionality for handling data streams is implemented by the DataOnStream component that allows data to be *pushed* to the AMIDST system (in contrast to a pull-approach that is standard when dealing with static database). To allow for variation in the run-time performance of the system a buffer component (Buffer) is needed for storing the most recent unprocessed cases. Each of the data sources are furthermore connected to the Data Instance component. This component consists of a single class that can represent a particular evidence configuration, such as the observed values of a collection of variables at time t or a particular row in a database.

Implementations of structural learning algorithms will be realized through the Structural learning component and its specialized sub-classes. The design currently includes components for supporting PC and TAN learning in a parallel setting (cf. Task 4.1). Currently, the implementation supports standard PC learning and parallel TAN learning by interfacing to the Hugin API.

The parameter learning approach will be realized through the Parameter learning component. This component is intimately connected to the Inference component, which we plan to implement in two forms: variational message passing (implemented in the VMP component) and expectation propagation (implemented in the EP component). These two inference methods both support a Bayesian approach to parameter learning and are directly linked to the activities in Task 4.2 and Task 4.4. The design of this part of the framework is flexible in the sense that it easily accommodates potential future learning-based extensions of the framework, e.g., maximum likelihood learning based on the expectation maximization algorithm (see Section 3).

Variable selection is handled in the corresponding component in Figure 5, which is connected to both the model component (PGM) and the Database component.

The high-level design description above provides an overview of the current design of the AMIDST software framework with particular focus on the components that are directly related to model learning. The current status of the software implementation in relation to this design is as follows:

- The core components (marked as blue in Figure **??**) has been implemented. This includes data structures for variables, graphs, Bayesian networks, dynamic Bayesian networks, key distributions such as multinomial and conditional linear Gaussian distributions represented in both standard form as an exponential family.
- Components defining data source management functionalities have been implemented, which includes support for handling static database (on disk and in memory) as well as streaming data.
- Methods for transforming AMIDST models to and from Hugin has been implemened based on the Hugin API.

Please see the appendix for the class diagrams for the implemented components.

Remark: The implementation is still ongoing and will continue so until the deliverable is submitted. Thus I expect that we will be able to expand the above list a bit. In particular, I know that there is currently work going on to provide parallel TAN learning in the AMIDST toolbox by calling the Hugin API. This would also be demonstrated at the review meeting.

5 AS – Task 4.1: Parallelization of structural learning

This section will give updates from Task 4.1, with focus on the actual tool-box implementation of parallel PC. We have two sources of information/ideas how to proceed:

- The slideset/poster we discussed some time ago.
- Hugin people are investigating lines for parallelizing the PC using multi-thread and relying on the TAN-PGM paper.

We should also discuss the learning of TAN classifiers (the PGM paper). Anders has suggested to Antonio that we could include the PGM paper here (except maybe the experiments, that could be reported in D5.1). In this way there would be no rush for having it implemented in the toolbox.

The setup of this section is that we show in practice how the initial procedures described in the design section is to be used.

6 HL – Initial results from the other tasks

[[**This section will discuss any results or ideas that have been developed in Tasks 4.2, 4.3, 4.4. Not much to include. Maybe delete?**]]

7 Conclusion

This document summarizes the progress of the software development related to learning functionality in the AMIDST toolbox. Parameter learning functionality rests heavily upon efficient design and implementation of core components and inference engines, and these parts were therefore discussed in some detail. Structural learning is ongoing (in Task 4.1) and will be finalized in Month 15. A preliminary discussion of the results was given. Implementation of dedicated functionality for parameter learning (Em algorithm and similar) has not yet started, and is not discussed.