International Journal on Artificial Intelligence Tools

B-COURSE: A WEB-BASED TOOL FOR BAYESIAN AND CAUSAL DATA ANALYSIS

PETRI MYLLYMÄKI, TOMI SILANDER, HENRY TIRRI, PEKKA URONEN

Complex Systems Computation Group (CoSCo)
Helsinki Institute for Information Technology
P.O.Box 9800, FIN-02015 HUT, Finland
http://cosco.hiit.FI/

Received February 11, 2002
Revised (revised date)

B-Course is a free web-based online data analysis tool, which allows the users to analyze their data for multivariate probabilistic dependencies. These dependencies are represented as Bayesian network models. In addition to this, B-Course also offers facilities for inferring certain type of causal dependencies from the data. The software uses a novel "tutorial style" user-friendly interface which intertwines the steps in the data analysis with support material that gives an informal introduction to the Bayesian approach adopted. Although the analysis methods, modeling assumptions and restrictions are totally transparent to the user, this transparency is not achieved at the expense of analysis power: with the restrictions stated in the support material, B-Course is a powerful analysis tool exploiting several theoretically elaborate results developed recently in the fields of Bayesian and causal modeling. B-Course can be used with most web-browsers (even Lynx), and the facilities include features such as automatic missing data handling and discretization, a flexible graphical interface for probabilistic inference on the constructed Bayesian network models (for Java enabled browsers), automatic pretty-printed layout for the networks, exportation of the models, and analysis of the importance of the derived dependencies. In this paper we discuss both the theoretical design principles underlying the B-Course tool, and the pragmatic methods adopted in the implementation of the software.

Keywords: Bayesian networks, causal networks, model selection, probabilistic inference, interactive tutorials, ASP

1. Introduction

B-course is a free* online data (dependency) analysis tool motivated by the problems in the current practice in statistical data analysis. In many cases, when practitioners in various fields apply analysis tools, the underlying assumptions and restrictions are not clear to the user, and the complicated nature of the software encourages the

*The B-Course service (http://b-course.hiit.fi or http://b-course.cs.helsinki.fi) can be freely used for educational and research purposes only.

users to a "black box" approach where default parameter values are used without any understanding of the actual modeling and analysis task. This observation holds both in scientific data analysis (e.g., in social sciences) and in "business" data analysis, where the users are not experts in the data analysis methods underlying the analysis software. This has lead to the situation where the conclusions derived from an analysis are frequently far from the intended plausible reasoning.

The B-Course tool is implemented as an Application Service Provider (ASP) — an architectural choice we feel is very natural in the context of data analysis. There is no downloading or installation of software, ASP allows a thin client at the user end and the computational load for searching models is allocated to a server farm. B-Course can be used with most web-browsers (even Lynx), and only requires the user data to be a text file with data presented in a tabular format typical to any statistical package (e.g., SPSS, Excel text format).

From the methodological point of view, B-Course is an attempt to offer a multivariate modeling method that can also be understood by applied practitioners. It is a first step in this direction and consequently can definitely be improved upon. However, it makes a serious attempt to give an informal introduction to the approach adopted. The software uses a novel "tutorial style" user-friendly interface which intertwines the steps in the data analysis with support material that gives an informal introduction to the Bayesian approach. Thus the analysis methods, modeling assumptions and restrictions are totally transparent to the user. However, this transparency is not achieved at the expense of analysis power — with the restrictions stated in the support material, B-Course is a powerful analysis tool that can be expanded to address some of its current limitations. B-Course supports inference on the constructed Bayesian network model as well as exporting the model for further use.

One of the design choices for B-Course was to adopt the Bayesian framework as indicated by the fact that the dependency models constructed are represented by Bayesian networks. We have chosen the Bayesian modeling framework, since we find it easier to understand than the classical statistical (frequentist) framework, and from our experience it seems that it is more understandable to the users also. We also feel that it has benefits over the classical framework, avoiding some of the anomalies caused by the hidden assumptions underlying the standard methods developed decades ago. This is not to say that Bayesian approaches do not have problems of their own — both theoretical and practical problems are lively discussed in the literature [1,4,22,10,15,14,19].

We strongly believe that Bayesian dependency modeling is a valuable tool in practitioner's data analysis toolbox. However, B-Course concentrates on being understandable, not merely being state-of-the-art. Almost all parts of B-Course can and will be improved upon — in our initial design we have favored simplicity and elegance over possibly minor gains in performance. On the other hand, presently there are not many tools around to do dependency modeling even at B-Course's current sophistication level. In addition to being an integrated service coherently implementing many of the methods resulting from research by us and others during the years, B-Course has also several unique features not available in any other software we are aware of.

In this paper we discuss both the design principles of B-Course, and methods adopted in the implementation of the software. We begin by discussing the problem addressed by B-Course, i.e., general principles of dependency modeling with Bayesian networks (Section 2). In Section 3 we then proceed by demonstrating a simple walk-through of a typical data analysis session. In addition to this discussion, we strongly encourage the reader to experiment with B-Course by using the "ready-made trails" provided by the service, or possibly with their own datasets if available. Results of preliminary systematic empirical validation tests with B-Course can be found in Section 5. Section 6 concludes our discussion.

2. Dependency Modeling with Bayesian Networks

In our context, dependence modeling means finding the model of the probabilistic dependences of the variables. In dependency modeling one tries to find dependencies between all the variables in the data. Since we are using probabilistic models, in more technical terms this means modeling the joint probability distribution. Dependencies can also be used to speculate about causalities that might cause them. Besides revealing the domain structure of the data, dependency models can be used to infer probabilities of any set of variables given any (other) set of variables. This will lead to a "game" where one can interactively study the model by probing it as implemented by the inference part in the B-Course software.

For the above purposes, in B-Course one will only need something that is called pairwise conditional dependencies, since that is the only type of dependency[†] that appears in our models. Saying that variables A and B are dependent on each other means that if one knows what the value of variable A is, it helps to guess what the value of variable B is.

To illustrate the type of models B-Course searches for, let us look at a small example. For our present purposes it is not necessary to study the models in great detail, the example just tries to give an idea about the dependency models. So let us assume that our model has four variables A, B, C and D. In Table 1 we list a set of statements about dependencies. This type of a set of statements defines a *dependency model* (let us call the example dependency model M_1). Obviously, if the set of dependencies is large, such a descriptive list representation becomes impractical and hard to understand.

In its full generality the problem of finding the best dependency model in an arbitrary set of models is intractable (see the discussion in [22,4]). In order to make the task of creating dependency models out of data computationally feasible, B-Course makes two important restrictions to the set of dependency models it considers.

[†]In the following, "dependency" always means "pairwise conditional dependency". It should be observed that this notion should not be confused with pairwise correlation.

Table 1. An example of a dependency model.

- *A and B are dependent on each other if we know something about C or D (or both).*
- *A and C are dependent on each other no matter what we know and what we don't know about B or D (or both).*
- *B and C are dependent on each other no matter what we know and what we don't know about A or D (or both).*
- *C and D are dependent on each other no matter what we know and what we don't know about A or B (or both).*
- *There are no other dependencies that do not follow from those listed above.*

Firstly, B-Course only considers models for discrete data and it discretizes automatically all the variables that appear to be continuous. Secondly, B-Course only considers dependency models where the list of dependencies can be represented in a graphical format using Bayesian network structures [20,17,9]. For example, the list of dependencies in Table 1 can be represented as a Bayesian network in Fig. 1.

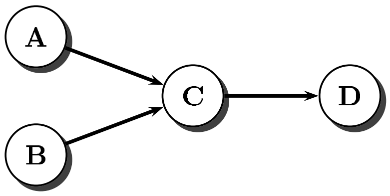

Fig. 1. A Bayesian network representing the list of dependencies in Table 1.

An important property of Bayesian network models is that the joint probability distribution over the model variables factorizes to a product of n conditional probability distributions:

$$P(X_1, \ldots, X_n) = \prod_{i=1}^{n} P(X_i|\Pi_i), \quad (1)$$

where Π_i denotes the parents (the immediate predecessors in the graph) of variable X_i. Consequently, the parameters θ of a Bayesian network model M consist of probabilities of the form $\theta_{ijk} = P(X_i = x_k|\Pi_i = \pi_j)$, where π_j denotes the jth value configuration of the parents Π_i. In the sequel we will assume that the reader is familiar with the basics of Bayesian networks, and refer to the introductions/textbooks in the literature (see, e.g., [12]).

This subset of models is interesting, but it has its limitations too. More specifically, if the variables of our model are in causal relationships with each other, and if in our domain there are no latent variables (i.e., variables that for some reason are not included in our data) that have causal influence on the variables of our model, then the dependencies caused by these causal relationships can be described by a Bayesian network. On the other hand, latent variables often induce dependencies, that cannot be described accurately by any Bayesian network structure. That can severely restrict our ability to automatically infer something about causalities just based on statistical dependencies (see Section 4.7).

Using the Bayesian approach provides for a way to recognize a good model when the software finds one: in the Bayesian framework a good dependence model is one with a high probability. Notice that it takes a Bayesian approach to speak about the probability of the dependencies. Further discussions on this topic is deferred to Section 4.2.

3. Walking through B-Course

In addition to offering a tool for automatically building dependency models that lead to (unsupervised) probabilistic models of the joint probability distribution, the new version of B-Course offers also a possibility for building (supervised) classification models. This "C-trail" of B-Course will however not be discussed in this paper — in the following we focus on the unsupervised "D-trail".

The B-Course "D-trail" data service offers a simple three step procedure (data upload, model search, analysis of the model) for building a Bayesian Network dependency model. As B-Course is used via a Web-browser, user can freely use the browser features ("Back" and "Forward" buttons, resizing of the window etc.) during this procedure. In particular, a first time user is encouraged to follow the links leading to pages explaining many of the concepts discussed in the interface. These pages form the "B-Course library" that is maintained and updated every time a new feature is added to the analysis.

Of the three main steps, the last one is the most complex one as it allows the user to interactively use the inferred model, export both the graphical and textual representations of the model, check for strengths of the dependencies etc. It should be emphasized that there are no parameter settings involved except the fact that the user decides the length of the search phase by interactively inspecting the progress of search. Discretization, handling of missing data, setting non-informative priors and other technical details are handled automatically by the software. In the following we give a short description of each of these main steps.

3.1. *Step 1: Data upload*

B-Course attempts to give a simple, yet accurate description of the format of the data it accepts. In general it expects the data to be in tab-limited ASCII format with additional header line containing the names of the variables. This format is readily available in most of the database, spreadsheet and statistical software. B-Course also allows for missing data.

Uploading the data is implemented by standard HTML-form File input, that sends the data file to the server. As B-Course is currently implemented by using a server farm, the front end server directs the data to one of the servers in B-Course server pool in order to do load-balancing.

B-Course notifies user of the possible problems during the data upload. It also gives simple descriptive statistics of each variable so that the user can verify the upload was successful. At this point user can also exclude the variables he/she does not want to be part of the model, such as data ID etc. (see Fig. 2).

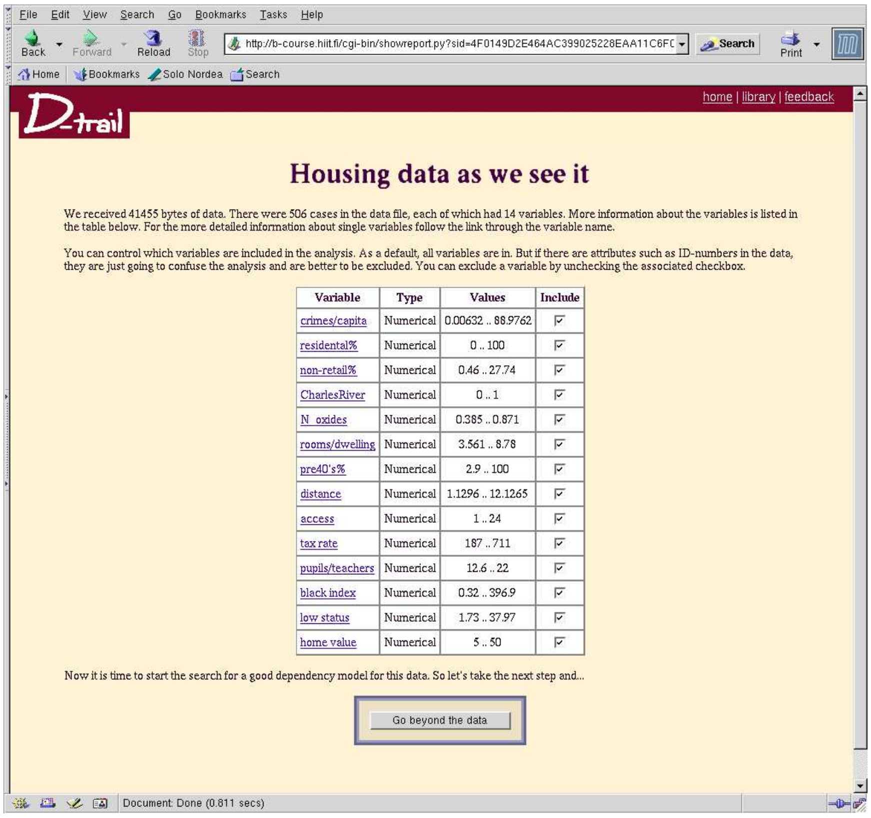

Fig. 2. Summary information presented by B-Course after uploading the data. A click on the variable name provides for descriptive statistics of the variable in question.

3.2. *Step 2: Model search phase*

In the model search phase the user is first prompted to initiate the dedicated server to start searching for a good Bayesian network model for the data. Once the search is on, the user is lead to a page showing the current best model. User can now study the structure of this model, but she can also ask for an updated report on the search. B-Course then again shows the current best model together with a report how much better the current model is compared to the previous one (assuming some improvement has occurred). The search can be stopped any time — for example, when no progress in search has been gained for some time — or the user can wait until the system search time limit (currently 15 minutes) has been reached. Searching for the dependency model is computationally very intensive (the problem is NP hard) and in any realistic case with many variables it is impossible to search exhaustively the whole search space.

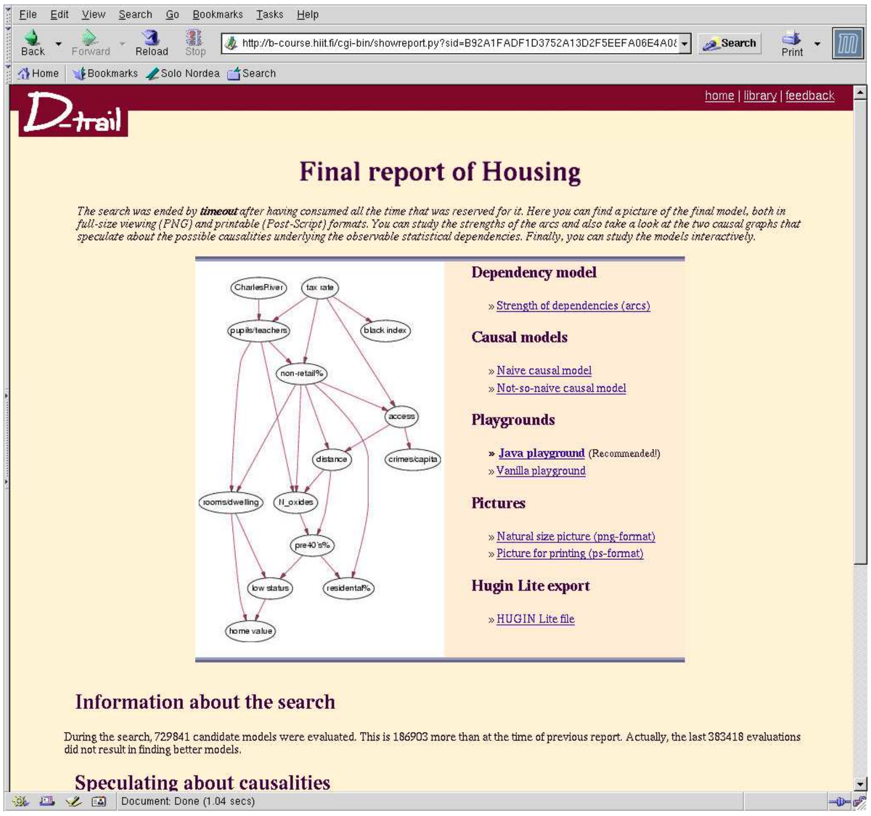

Fig. 3. Search status report after the model search is completed (shows search statistics and the best dependency structure found).

3.3. *Step 3: Analysis of the model found*

Once the search has ended, B-Course gives the final report together with a list of ways to study the selected dependency model (see Fig. 3). The final report displays the constructed graph structure which the user can save if needed. The user is also given a report on strengths of the pairwise unconditional dependencies (i.e., arcs in the constructed Bayesian network) of the model. In addition to the standard

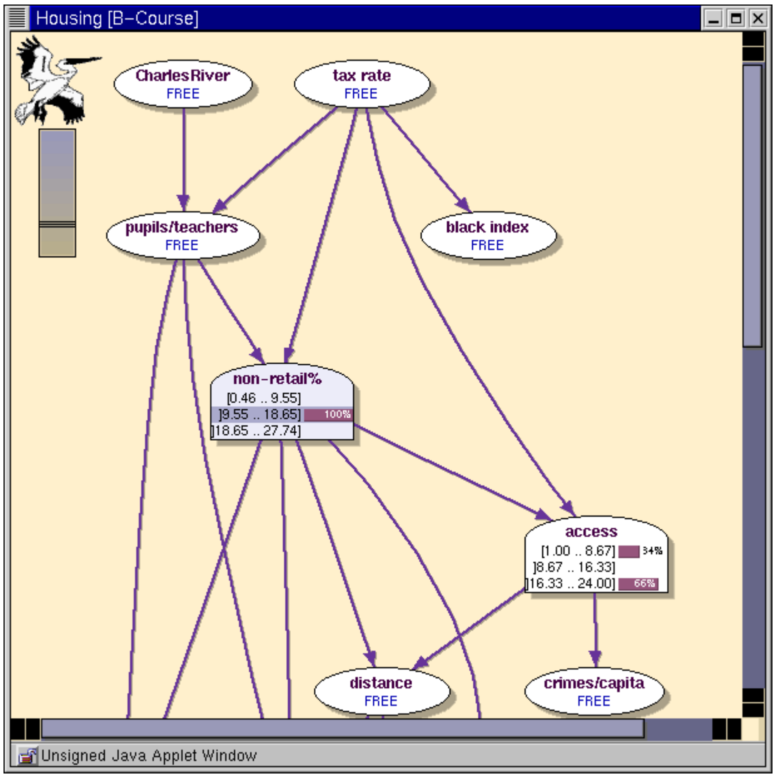

Fig. 4. Probabilistic inference can be performed on the constructed Bayesian network model with the inference Java applet.

Bayesian network representation, B-Course also offers two graphical representations describing the possible causal relationships that may have caused the dependencies of the model. These causal graphs are based on the calculus introduced by Pearl [21]. As far as we know this feature is unique to B-Course, we are not aware of any other software package supporting Pearl's causal analysis.

B-Course also provides interactive tools called "playgrounds" that allow user to perform inference on the constructed Bayesian network. Several playgrounds are offered in order to support browsers with various capabilities. The "Vanilla playground" is intended to be used with "low-end" browsers with restricted graphical capabilities and works even with text-only browsers such as Lynx. The "Java playground" (see Fig. 4) requires a Java-enabled browser, but then offers a more flexible graphical user interface with zooming, pop-up displays of the distributions attached to variable nodes etc. In addition to using B-Course playgrounds online for model inspection, the model can also be exported in a format accepted by Hugin-software in order to allow off-line use of the model.

4. The B-Course Approach

One of the most important advantages of the Bayesian modeling approach is that it offers a solid theoretical framework for integrating subjective knowledge with objective empirical observations. In the current version of B-Course we however neglect this opportunity, and aim to be as objective as possible. Technically this means that the prior distributions used should be as non-informative as possible. The reason for this choice is that one of the main design principles underlying the current version of B-Course was that it should be easy to use also by non-experts. This implies that the user cannot be expected to be able to enter complex technical parameters or make decisions on selection of the mathematical methods used. Consequently, B-Course has no user definable technical parameters — all the data preprocessing (discretization, missing data handling etc.) and search related decisions (search criteria, search bias etc.) are handled automatically. From the user point of view, part of this simplicity is due to the elegant Bayesian theory that offers a solid theoretical background with very few parameters to tamper with, but some of it is based on hardwired implementation choices that are made on the behalf of the user. However, following the transparency requirement, B-Course tries to be very explicit about these choices in its online documentation. In the following we will discuss these implementation choices, occasionally also touching the relevant issues in Bayesian modeling.

4.1. *Discretization*

The set of dependency models B-Course considers (namely discrete Bayesian networks) expects the variables to be categorical (e.g., gender, favorite color etc.). However, many times the variables are naturally numerical (like age) or the values have at least some natural order (like the so-called Likert scale “strongly agree”, “agree” “indifferent” “disagree” and “strongly disagree”). In such cases B-Course will categorize the variables, a process which destroys all the information about the numerical value and order. Continuous numerical variables are discretized into intervals and even the order of the intervals is neglected. Similarly in ordered variables the information about the order is ignored.

The main reason for such a discretization is that for categorical variables we can build models that capture “non-linear” relationships between variables. We also get rid of the restricting distributional assumptions (like multivariate normality assumptions prevalent in current statistical practice). Thus the advantage is making less assumptions and the possibility to find more complex relationships.

However, discretization is also in many ways problematic. We loose statistical power, since if the relationship between variables happen to be linear, linear models will find out that relationship with less data than B-Course will (which is not surprising since linear models are naturally good in detecting linear dependencies, otherwise they would be totally useless). On the other hand, as opposed to many classical statistical estimation procedures, no Bayesian analysis is ever nonviable due to "too little data". In other words, the Bayesian analysis takes into account all the data available, there are no preset sample sizes that have to be satisfied in order to be able to perform the dependency analysis. If the database is small, the dependencies found are simply weaker and the best model found may not be very much better than the second best.

How to discretize numerical data so that the amount of information lost is as small as possible? The canonical Bayesian answer could be that good discretization is a discretization that is very probable. However, it is not clear that discretization can be handled as something we do not know, but that still is there. Rather it is an artifact that has been created, so the probability of discretization is not necessarily a meaningful concept. At the moment B-Course does not attempt the sophisticated full Bayesian approach, but instead takes a straightforward approach and tends to discretize the data to very few intervals. This way the amount of data required for finding out interesting dependencies can be expected to be reasonably small.

If the user has continuous variables he/she might want to manually discretize the data beforehand to get a meaningful discretization. Many times certain discretization is meaningful because of some theoretical assumptions in the domain. Naturally such things cannot be inferred automatically. However, it should be noted that the way one discretizes the data can make a notable difference when building the model. If one discretizes continuous variables to just few intervals, one is likely to find out more dependencies than if the values are discretized to very many intervals. In addition, the result may change if one changes the division points that define the discretization.

4.2. *Quality of the model*

B-Course adopts a standard Bayesian answer to assess the quality of the model: the model that is most probable for the data is selected to be the best one [4,1]. While this principle is both intuitive and theoretically justified, it is not the only possible one. For example, it would be very legitimate to base the quality score of the model on some kind of usefulness measure rather than probability. However, many times this 'usefulness' is very hard to formalize and there are no standard ways to express this type of information. Incorporating such features to the software would also make B-Course harder to use.

Given a data set D, the most probable dependency model $\hat{M}$ is the one maximizing the posterior probability:

$$\hat{M} = \arg\max_{M} P(M|D) = \arg\max_{M} P(D|M)P(M). \tag{2}$$

The last equality follows from the fact that with respect to the dependency models M, the probability $P(D)$ is a constant that can be ignored when comparing the probabilities of models.

Following the objectivity requirement discussed above, in the current version of B-Course, all models M are assumed to be equally probable before any data is seen. In other words, we assume that the prior distribution $P(M)$ over the models is uniform, i.e., $P(M_i) = P(M_j)$ for any two models M_i and M_j. This means that posterior probability of the models is now proportional to the *marginal likelihood* of the data:

$$P(M|D) \propto P(D|M) = \int P(D|\theta, M)P(\theta|M)d\theta, \tag{3}$$

where the integral goes over all the parameter instantiations θ of model M.

As was seen in Section 2, the parameters of a Bayesian network model M consist of probabilities of the form $P(X_i = x_k|\Pi_i = \pi_j)$, where Π denotes the parents (the immediate predecessors in the graph) of variable X_i. The data D is discrete (originally discrete, or discretized as discussed above), so it is natural to treat the data as a multinomial sample with sufficient statistics $N_{ijk}, i = 1, \ldots, n, j = 1, \ldots, q_i, k = 1, \ldots, r_i$, where N_{ijk} is the number of rows in D where variable X_i has value x_k and the parents Π_i of X_i have a value configuration π_j.

The prior $P(\theta|M)$ for the parameters is assumed to be the Dirichlet distribution that is a conjugate prior distribution for the multinomial. This assumption, together with certain additional technical assumptions (see e.g. [13]), allows us to compute the marginal likelihood term in Eq. (3) in closed form:

$$P(D|M) = \prod_{i=1}^{n} \prod_{j=1}^{q_i} \frac{\Gamma(N'_{ij})}{\Gamma(N'_{ij} + N_{ij})} \prod_{k=1}^{r_i} \frac{\Gamma(N'_{ijk} + N_{ijk})}{\Gamma(N'_{ijk})}, \tag{4}$$

where Γ denotes the gamma function, n is the number of variables in M, q_i is the number of value configurations for the parents of variable X_i, r_i is the number of values of X_i, N_{ijk} are the sufficient statistics discussed above, and $N_{ij} = \sum_{k=1}^{r_i} N_{ijk}$. The constants N'_{ijk} are the *hyperparameters* determining the prior distribution $P(\theta|M)$.

4.3. *Choosing the prior distribution for the parameters*

Eq. (4) gives us now the required model quality criterion for validating a model M, as soon as we choose the hyperparameters N'_{ijk} determining the parameter prior $P(\theta|M)$. Following again the objectivity principle, the hyperparameters should be selected in such a way that the resulting prior distribution is as non-informative as possible. Setting N'_{ijk} to one for all triplets i, j, k produces the uniform prior distribution, which however — perhaps rather counter-intuitively — can not be regarded as a very good non-informative prior. One reason for this is that the uniform distribution is not invariant to variable transformations. A practical consequence of this fact is that with the uniform prior, the marginal likelihoods of two Bayesian network models representing the same set of dependency statements may be different. All in all, unfortunately it turns out that the problem of choosing a non-informative prior is a most controversial issue that has raised a lot of discussion [2,8], and that no simple solution can be found for solving this problem.

One solution, suggested in [5], is to set $N'_{ijk} = N'/(r_i q_i)$, where N' is a global constant called the *equivalent sample size (ESS)*. The intuition behind the equivalent sample size prior is that it effectively behaves as if it were calculated from a "prior" data set of size N'. An important advantage with this prior is that the marginal likelihood score (4) now follows the *score equivalence* criterion: if two Bayesian network models M_1 and M_2 represent the same set of dependency statements, then the marginal likelihoods are also equivalent.

The ESS prior described above still leaves us with one parameter, N'. In order to determine this constant, let us have a look at another prior with nice theoretical properties, the so called *Jeffreys' prior* [16]. When Jeffreys' prior is proper, it is defined by [1,23]:

$$\pi(\theta) = \frac{|I(\theta)|^{1/2}}{\int |I(\eta)|^{1/2} d\eta}, \tag{5}$$

where $|I(\theta)|$ is the determinant of the *Fisher (expected) information matrix*. One of the advantages of Jeffreys' prior is that unlike with the uniform prior, the prior distribution is now invariant with respect to one-to-one transformations in the parameter space. Additional theoretical properties of this function are studied in [1,7,23].

In the Bayesian network model family, Jeffreys' prior turns out to be proper: the formulas for computing Jeffreys' prior for Bayesian networks can be found in [18]. However, unfortunately the resulting distribution is not of the desired conjugate Dirichlet form, which makes this prior computationally difficult to use.

The basic strategy in the current version of B-Course is to try to choose the parameter N' in such a way that the resulting ESS prior distribution is close to Jeffreys' prior. The hope is that this would give us a prior that is not too sensitive for variable transformations, but is still computationally convenient to use. In practice we currently set $N' = (\sum_{i=1}^{n} r_i)/2n$. The motivation for this is that as can be seen from the result given in [18], with an empty network (with no arcs), this selection produces exactly the Jeffreys' prior.

4.4. *Handling missing data*

Missing data is a problem for many statistical procedures, and B-Course is not an exception. Bayesian theory has a very clear theoretical answer to the lack of information caused by missing data. Unfortunately, this answer is computationally infeasible, so B-Course ends up doing something much simpler to approximate this answer.

The simplest way to handle missing data is to ignore all the data rows that have some missing entries. The other possibility is to impute the missing data. That means guessing some values to those entries of the data matrix that are missing. After imputation we could then continue the analysis as if there were no missing data at all. B-Course does something between these two extremes. It tries to throw away only those parts of the data row that are missing. In fact for technical reasons B-Course discards a little bit more than this.

B-Course uses the method called "ignoring". The calculation of the probabilities of the models is essentially based on the frequency of different patterns of data in the database. When calculating these pattern frequencies B-Course simply ignores the patterns that contain missing data. Because of this, the probability of the model is slightly miscalculated. This method works fine when the data is missing randomly. However, in many cases values are not missing completely at random. Since B-Course will deal with discrete values anyway, it is often a good idea to handle missing values as legitimate values for a variable. In particular, this should be done, if we suspect that there is some systematic reason for values to be missing. Of course, handling missing values as "ordinary" values is not meaningful, if the amount of missing values is very small (like once in one variable). To treat missing data as value of its own the user can simply replace missing positions with any name he/she likes (e.g., ?, missing, *, no answer, etc) as long as the newly created value name does not clash with existing values. Of course, user is also free to use any other way to pre-fill the missing data before uploading the data into B-Course.

4.5. *Model search*

In the full Bayesian approach, in making predictions one needs to marginalize over the complete posterior distribution of models [4,1]. In contrast to this, B-Course attempts to find and use a single best dependency model structure — in standard statistical terminology B-Course is doing model selection. The main reason for this is that the full posterior distribution is extremely complex, and thus not computationally feasible to use. Furthermore, from the data mining point of view it is not clear at all how the full posterior could be meaningfully presented to the user.

Finding the most probable model for the (discrete) data (even without missing values) is NP-hard [6], so the optimal model selection method would be to go through all the models, calculate the probability for each of them and then pick the best one. However, the number of Bayesian network structures grows very rapidly as a function of the number of variables [24]: for example, for 20 variables the severely underestimated number of possible Bayesian network structures is $1.6 * 10^{57}$, hence in practice we are forced to use search heuristics. B-Course uses a combination of stochastic and greedy search heuristics to explore the very high-dimensional spaces. Notice that since the model quality criterion and the search procedure have been separated, the B-Course "search engine" can be improved independently of the model selection criterion.

4.6. *Weights of dependencies*

The adopted model selection approach has naturally some drawbacks: when there are many models that have approximately the same probability (or marginal likelihood) as the most probable model, those other models should also be consulted when we make predictions (generalizations) for the data we have not seen, i.e., when we want to say something about cases that were not in our data sample. In fact to be exact, all the possible models should be used when making predictions and the contribution of each model should be proportional to model's probability. If we pick just one model, our predictions (generalizations) are worse than if we use many models. Only in cases where the most probable model has very much higher probability than any other model, other models can be neglected and the predictions can be done with the most probable model. This is natural, since in this case we are almost sure that our model is the proper one. In addition, using just a single model does not allow us to estimate our certainty about the predictions that are based on that model. This is due to the fact that picking one model equals pretending that we have found the "true" model of the world, and when the truth is known, there is no uncertainty left. Of course, if it so happens, that the best model has the probability very near one, we are safe. In future versions of B-Course this issue will be at least partially resolved by showing a high-scoring set of dependency models for the user, instead of showing only the single best model.

In the current version of B-Course, uncertainty about goodness of the network found during the model search is expressed by analyzing the importance of individual arcs. However, although it is natural to ask how probable or strong certain unconditional dependency statement represented by an arc in the model is, in non-linear models it is not an easy task to give "strengths" to arcs, since the dependencies between variables are determined by many arcs in a somewhat complicated manner. Furthermore, the strength of a dependency is conditional on what you know about the other variables.

Fortunately one can get a relatively simple measure of the importance of an arc by observing how much the probability of the model is changed by removing the arc. This type of a local analysis can be motivated by the following line of reasoning: The final model is the most probable model B-Course could find given the time used for searching. However, there may be other models that are almost as probable as our final model. Natural candidates for other probable models are the ones that can be obtained by slightly changing the final model by removing one of the existing arcs.

For studying the importance of the arcs, B-Course offers a list of statements describing how removing an arc affects the probability of the model. If the removal makes the model much worse (i.e., less probable), it can be considered as an important arc (dependency). If removing the arc does not affect the probability of the model much, it can be considered to be a weak dependency. In the list the strongest dependencies are listed first. For "weaker" arcs A→B, B-Course also gives the probability ratio 1 : N that should be read as follows: the final model is N times as probable as the model that is otherwise identical, but in which the arc between A and B has been removed.

4.7. *Inferring causality*

The theory of inferred causation (see [21]) makes it possible to speculate about the causalities that have caused the dependencies of the model. In B-Course there are two different speculations (called "naive model" and "not so naive model") which are based on different background assumptions.

Inferring causality from statistical dependence is an issue of a long-term debate [25,11,21]. However, causal models have many desirable properties, so the task is worth pursuing in dependency modeling tools such as B-Course. It also appears that by making some additional assumptions, inferring causalities from observed statistical dependencies can be justified.

Arguing that we can infer causality from statistical dependencies necessarily relies on the properties of causality and statistical dependencies. While people seem to be naturally good in inferring about causality, they are not naturally talented in making inferences about statistical dependencies. In general it is somewhat plausible to think that all the dependencies between things in the world are due to some kind of a causal mechanism. In the "naive model" B-Course makes an additional assumption that all the dependencies between variables are due to the causal relationships between variables in the model. Effectively this denies the possibility that one has excluded some variables that could cause dependencies in our model. That equals denying the possibility of latent causes. Naive model also assumes that there are no causal cycles in the domain.

If we, however, make the naive assumption of excluding latent variables, the inference of causes seem to become possible since all the unconditional dependencies between A and B (A and B are dependent of each other no matter what we know or don't know about other variables) must be explained by either A causing B or B causing A. But how can we know the direction of causality? We cannot always, but sometimes we are lucky to have such a model, that the coexistence of dependencies cannot be explained without a certain causal relationship. If A and B seem to be dependent no matter what, and B and C seem to be dependent no matter what, but A and C are not dependent of each other if we know something about certain other variables S (S can be empty), and nothing about B and the rest of the variables, then we know for sure that A has causal effect on B. How come? This is because B cannot be the cause of A or otherwise A and C would always be dependent too, but we just said that that sometimes (given S not containing B) they are independent.

Sometimes inferring existence or non-existence of causalities between variables is possible even if we relax the naive assumption that there are no latent variables involved in dependencies. In general all the dependencies between variables might be caused by one latent variable. Postulating such variable is however somewhat against scientific inquiry where the goal is to make as few assumptions as possible (Occam's razor). If we however restrict ourselves to the latent variable models where every latent variable is a parent of exactly two observed variables, and none of the latent variables has parents, we can infer something about causal relationships of observed variables. We call this restricted set of latent variable models "not so naive causal models". This restriction is not as bad as it sounds, since it can be shown, that under very reasonable assumptions, all causal models with latent variables can be represented as models in this class.

Sometimes the subset of dependencies in our model can help us exclude the possibility of A causing B even when A and B seem to be always dependent. This is the case if there is a third variable C that given S (that does not include A or B or C) is dependent of A but independent of B. If A were a direct cause of B, the dependence between C and A would always make C and B dependent too. This is against our assumption that our model contains a statement saying C and B are independent given S (that does not contain A or B or C). The only possibilities left are that either B causes A or that there is a latent common cause for A and B, that makes them appear dependent.

The "naive" and "not so naive" approaches for causal modeling have both been implemented in the B-Course software, and the causal dependencies found are visualized as a graph. Although B-Course is to our knowledge the first software package supporting the causal modeling framework described in [21], unfortunately the current version of the software supports only causal analysis of data — causal inference can not yet be performed on top of the constructed causal models in the same way probabilistic inference can be performed on the constructed Bayesian network models. The work in this area is still in progress.

5. Empirical Validation

When designing a learning algorithm to construct models from data sets with non-informative prior information (i.e., no preference for the structure in advance), a challenging and interesting task is to evaluate the "quality" of the learning algorithm. In addition to the theoretical justifications of the Bayesian model construction, with such an analysis tool as B-Course, empirical validation is a necessary step in the development process.

There are several possible ways to study the performance of a model construction algorithm, and many of the schemes are based on simulating the future prediction tasks by reusing the data available, e.g., with cross-validation methods. These approaches have problems of their own and they tend to be complex for cases where one is interested in probabilistic models for the joint distributions as opposed to for example classification. Therefore in many cases in the literature the so-called synthetic or "Golden Standard" approach is used to evaluate the learning algorithm. In this approach one first selects a "true model" (Golden standard) and then generates data stochastically from this model. The quality of the learning algorithm is then judged by its ability to reconstruct this model from the generated data.

In addition to the already quite extensive use with real data sets, B-Course was also tested using synthetic data sets generated from known Bayesian networks. In this case the particular interest in these experiments was to find out how well the B-Course learner can "recover" the Golden Standard network for Bayesian networks of varying complexity using data sets of different sizes. Following the dependency modeling aspects underlying B-Course, the main interest was in comparing the structural differences, not parameters. Several sets of tests were performed by varying the network size (5, 15, and 50 nodes) and the dependency structure complexity against the maximum of n^2 arcs (0%, i.e., all independent, 10%, and 40% of possible structural dependencies). In addition, the average node incoming/outgoing degree (i.e., the number of arcs pointing to or from a node) was varied (1, 3, and 5). Finally the construction rate (i.e., the "statistical power") was studied by varying the size of the generated data set from 100 to 10000 data vectors. The resulting Bayesian networks were compared to the generating network by comparing the skeletons, i.e., the underlying undirected structure, and the V-structures which together define an equivalence relation among the networks [20].

The results clearly validated that the model search in B-Course finds dependency structures present in the underlying data generating mechanism. For the small networks (5 nodes), regardless of the structure complexity in almost all the cases the correct network could be recovered with 100 to 1000 data vectors. Even if this was not the case, the differences in B-Course inferred network and the "true" dependency model were only 1–2 missing dependencies. Similarly the performance with 15 node random networks was comparable (typically 1–2 missing dependencies or 1 incorrect V-structure), albeit now the data set sizes needed to recover the network were typically 1000 as opposed to 100 sufficient for the smaller networks. As expected, when the network size was increased to 50 with a notable connection complexity (0.1 to 0.4), even the data set sizes of 10000 were sufficient to recover the generating structure only very approximatively. The typical amount of missing dependencies varied from 10% to 50%. However, one has to remember that the amount of data used for these cases is way too small for such complex models by any means (networks with more than 1000 arcs with 10000 to 15000 parameters!). The main purpose of the tests with larger networks was to find out, whether the model search produces "spurious dependencies", i.e., adds dependencies that only reflect the noise in the data. In this respect B-Course is extremely well-behaving — it almost never adds a dependency where there should not be one and prefers simpler models in the light of lesser amounts of evidence (i.e., smaller data sets).

6. Conclusions and Future Work

The two main design principles in building the current version of B-Course were transparency and ease of use. On one hand, we wanted to build a system where the modeling assumptions are explicitly visible so that applied practitioners can fully understand what the results of the analysis of their data mean without having to consult a statistics textbook. On the other hand, we wanted the data analysis to be fully automated so that the users would not have play with parameters the meaning of which would be clear only to modeling experts. These two requirements were met by adopting the Bayesian dependence modeling approach: according to our experience, the basic theoretical concepts in this probabilistic framework seem to be easier to understand than the concepts of classical frequentist statistics, and by using a series of explicitly stated assumptions, we were able to get rid of all the model parameters, leaving the user with a fully automated data analysis tool, as was the goal.

Nevertheless, although the initial goals of the B-Course development project have been met with the current version of the software, the current implementation leaves naturally room for many improvements. For example, the modeling assumptions allowing us to offer the users a "non-parametric" tool are not necessarily always very reasonable. Furthermore, the question of choosing an objective "non-informative" prior for the model parameters turned out to be a most complex issue, linking this superficially simple task directly to the most fundamental problems in statistics. On the other hand, as the chosen Bayesian approach offers an elegant framework for integrating subjective knowledge with empirical observations, it would be nice to be able to offer the more sophisticated users a possibility for expressing their expertise as a prior distribution on the dependency statements, or on the model parameters. Finally, it is evident that uncertainty on the result of the analysis should be expressed in a more elaborate manner than with the straightforward local analysis of the importance of the individual arcs. These issues will be addressed when developing future versions of the B-Course data analysis service.

Acknowledgements

The B-Course software is based on the work done by the Complex Systems Computation research group and its affiliates during the last five years. In particular the direct and indirect contributions of Peter Grünwald, Petri Kontkanen, Jussi Lahtinen, Kimmo Valtonen and Hannes Wettig are greatly appreciated. This research has been financially supported by the National Technology Agency, and the Academy of Finland.

References

[1] J.O. Berger. *Statistical Decision Theory and Bayesian Analysis.* Springer-Verlag, New York, 1985.

[2] J.O. Berger and J.M. Bernardo. On the development of reference priors. In Bernardo et al. [3], pages 35–60.

[3] J.M. Bernardo, J.O. Berger, A.P. Dawid, and A.F.M Smith, editors. *Bayesian Statistics 4.* Oxford University Press, 1992.

[4] J.M. Bernardo and A.F.M Smith. *Bayesian theory.* John Wiley, 1994.

[5] W. Buntine. Theory refinement on Bayesian networks. In B. D'Ambrosio, P. Smets, and P. Bonissone, editors, *Proceedings of the Seventh Conference on Uncertainty in Artificial Intelligence*, pages 52–60. Morgan Kaufmann Publishers, 1991.

[6] D.M. Chickering, D. Geiger, and D. Heckerman. Learning Bayesian networks is NP-hard. Technical Report MSR-TR-94-17, Microsoft Research, 1994.

[7] B.S. Clarke and A.R. Barron. Jeffrey's prior is asymptotically least favorable under entropy risk. *Journal of Statistical Planning and Inference*, 41:37–60, 1994.

[8] R. Cowell. On compatible priors for Bayesian networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(9):901–911, September 1992.

[9] R. Cowell, P.A. Dawid, S. Lauritzen, and D. Spiegelhalter. *Probabilistic Networks and Expert Systems.* Springer, New York, NY, 1999.

[10] A.P. Dawid. Prequential analysis, stochastic complexity and Bayesian inference. In Bernardo et al. [3], pages 109–125.

[11] C. Glymour and G. Cooper, editors. *Computation, Causation and Discovery.* AAAI Press/MIT Press, 1999.

[12] D. Heckerman. Bayesian networks for data mining. *Data Mining and Knowledge Discovery*, 1(1):79–119, 1997.

[13] D. Heckerman, D. Geiger, and D.M. Chickering. Learning Bayesian networks: The combination of knowledge and statistical data. *Machine Learning*, 20(3):197–243, September 1995.

[14] C. Howson and P. Urbach. *Scientific Reasoning: The Bayesian Approach.* Open Court, Chicago, 1993.

[15] H. Jeffreys. *Theory of Probability.* Clarendon Press, Oxford, 1939.

[16] H. Jeffreys. An invariant form for the prior probability in estimation problems. *Proc. Roy. Soc. A*, 186:453–461, 1946.

[17] F. Jensen. *An Introduction to Bayesian Networks.* UCL Press, London, 1996.

[18] P. Kontkanen, P. Myllymäki, T. Silander, H. Tirri, and P. Grünwald. On predictive distributions and Bayesian networks. *Statistics and Computing*, 10:39–54, 2000.

[19] R. Matthews. Faith, hope and statistics. *New Scientist*, 156(2109):36–39, 22 November 1997.

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

[21] J. Pearl. *Causality: Models, Reasoning and Inference.* Cambridge University Press, 2000.

[22] J. Rissanen. *Stochastic Complexity in Statistical Inquiry.* World Scientific Publishing Company, New Jersey, 1989.

[23] J. Rissanen. Fisher information and stochastic complexity. *IEEE Transactions on Information Theory*, 42(1):40–47, January 1996.

[24] R. Robinson. Counting unlabeled asyclic graphs. In C. Little, editor, *Combinatorial Mathematics*, number 622 in Lecture Notes in Mathematics. Springer-Verlag, 1977.

[25] P. Spirtes, C. Glymour, and R. Scheines, editors. *Causation, Prediction and Search.* Springer-Verlag, 1993.