Software for data analysis with graphical models

Wray L. Buntine
RIACS at NASA Ames Research Center
Mail Stop 269–2
Moffett Field, CA 94035–1000, USA
wray@kronos.arc.nasa.gov

H. Scott Roy
Heuristicrats Research, Inc.
1678 Shattuck Avenue, Suite 310
Berkeley, CA 94709-1631, USA
hsr@heuristicrat.com

Abstract

Probabilistic graphical models are being used widely in artificial intelligence and statistics, for instance, in diagnosis and expert systems, as a framework for representing and reasoning with probabilities and independencies. They come with corresponding algorithms for performing statistical inference. This offers a unifying framework for prototyping and/or generating data analysis algorithms from graphical specifications. This paper illustrates the framework with an example and then presents some basic techniques for the task: problem decomposition and the calculation of exact Bayes factors. Other tools already developed, such as automatic differentiation, Gibbs sampling, and use of the EM algorithm, make this a broad basis for the generation of data analysis software.

Introduction

This paper argues that the data analysis tasks of learning and knowledge discovery can be handled using graphical models. This meta-level use of graphical models was first suggested by Spiegelhalter and Lauritzen [41] in the context of learning probabilities for Bayesian networks. An extension of the standard graphical model is used here that allows this kind of learning to be represented. The extension is the notion of a *plate* introduced by Spiegelhalter[1]. Plates allow samples to be represented explicitly on the graphical model, and thus reasoned about. This makes data analysis problems explicit in much the same way that utility and decision nodes are used for decision analysis problems [38].

Consider, for instance, Figure 1. This presents a situation where a mixture model with hidden variable *class* is used for subsequent prediction of var_1 from var_2 and var_3. The part to the left of the parameters ϕ and θ is the graphical representation of a sample. The contents of the *plate* (the box

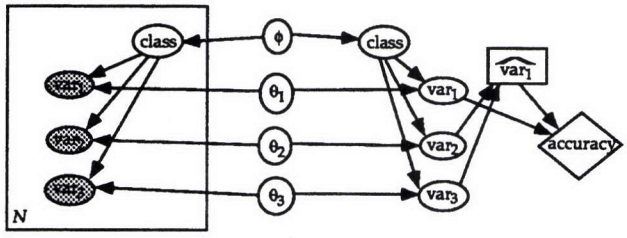

Figure 1: Simple unsupervised learning, with general prediction

around the nodes for *class*, var_1, var_2 and var_3) on the left indicates that a sample of N cases with variables var_1, var_2 and var_3 are given, while *class* is hidden, being unshaded. The plate indicates that its contained subgraph is replicated N times. The part on the graph to the right of the parameters ϕ and θ represents the prediction task. The value node on the right, the diamond, indicates that subsequent prediction accuracy is the goal of learning. Together, this graph indicates that the utility for the problem is $(var_1 - \widehat{var_1}(var_2, var_3))$, and the joint distribution of the parameters takes the form

$$
\begin{aligned}
&p(\phi, \theta_1, \theta_2, \theta_3, class, var_1, var_2, var_3, \\
&\quad class_i, var_{1,i}, var_{2,i}, var_{3,i} : i = 1, \ldots, N) = \\
&\quad p(\phi)\, p(\theta_1)\, p(\theta_2)\, p(\theta_3)\, p(class|\phi) \\
&\quad p(var_1|class, \theta_1)\, p(var_2|class, \theta_2)\, p(var_3|class, \theta_3) \\
&\quad \prod_{i=1}^{N} p(class_i|\phi)\, p(var_{1,i}|class_i, \theta_1) \\
&\qquad\qquad p(var_{2,i}|class_i, \theta_2)\, p(var_{3,i}|class_i, \theta_3)\ .
\end{aligned}
$$

There has been a recent push within the machine learning and neural network communities to dispel the magic and art from the various learning fields and present them more as engineering disciplines. Decision tree methods [5] and feedforward networks [30, 8] are some examples that show how already popular algorithms can be reengineered from well understood principles of probability in combination with the knowledge repre-

[1] Personal communication. The notion of a "replicated node" was my version of this developed independently. I have adopted the notation used by Spiegelhalter and others for uniformity.

sentation and standard search methods. A simple connectionist feed-forward network (using the notation of Hertz, Krogh and Palmer [23]) and its corresponding Bayesian network is given in Figure 2(a) and (b) respectively. Similarly, other neu-

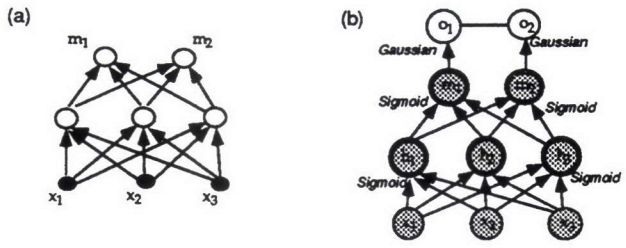

Figure 2: A simple feed-forward network: (a) in native form (b) as a DAG.

ral networks can be modeled with graphical models ("probabilistic networks").

This general approach, engineering using principles of probability, is now becoming widespread. The basic tools of probabilistic (Bayesian) inference used for this process are reviewed, for instance, by Tanner [42], Press [36], Kass and Raftery [28], Neal [34], and Bretthorst [2], and Madigan *et al.* [32]: various exact methods, Markov chain Monte Carlo methods such as Gibbs sampling, the EM algorithm, and the Laplace approximation. With creative combination, these are able to address a wide range of data analysis problems. Gilks, Spiegelhalter and Thomas have taken this process a step further by developing a compiler that generates Gibbs samplers from graphical specifications [19]. This handles a surprisingly broad number of statistical tasks [18].

It is the thesis of this paper that these techniques are now sufficiently well developed so that software support can be provided for their use in data analysis problems. That is, we are now able to generate components of data analysis algorithms, and even entire algorithms themselves from high-level specifications. The paper demonstrates the thesis by presenting a framework based around the use of graphical models as a specification language.

We begin with two examples. The first illustrates the intended use of the software we envisage, and the second gives some more mathematical detail. Then we outline in more detail the specification language we propose. Finally, we present some theoretical results necessary for developing the envisaged software. More details of these results can be found in [3], including results for deterministic nodes and techniques for doing differentiation, both used in modeling neural networks with probabilistic graphical models.

Two examples

The software we envisage is intended to be used in an iterative prototype-refine cycle using standard data manipulation and visualization packages such as Matlab, PV-Wave/IDL, or S-Plus. An important observation is that prepackaged data analysis software such as clustering, linear regression, and feed-forward neural networks are sometimes inadequate for the particular task at hand. While these packages are often good for exploratory data analysis, our experience and that of many others indicates that data analysis and knowledge discovery requires more flexibility in general. The first example below illustrates the kind of prototyping our envisaged software is intended to assist, and the second example illustrates some more of the mathematical detail.

Prototyping data analysis

This example will demonstrate how the system we propose would operate, reducing a problem that might require weeks of effort into an afternoon's work. Figure 3 plots the raw data for this example. The data give mean bid-ask prices posted by banks

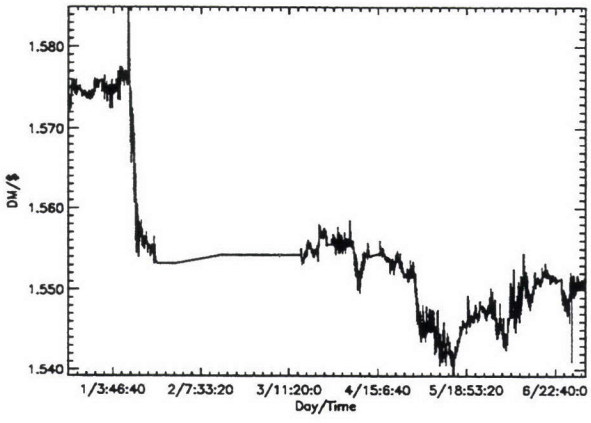

Figure 3: The mean bid-ask price for DM/$.

at various time points over the course of a week for turning dollars into Deutsch marks. The mean bid-ask price (average of the two) is a more stable indicator of the bank's pricing position because the bid or ask price alone also includes effects due to the banks policy on the bid-ask spread. Original data takes the form of a date and time, the bid and asking price, and the bank code.

Date	Bid	Ask	Bank
Sep 1 13:42:40	1.5737	1.5742	CONY
Sep 1 13:42:45	1.5735	1.5745	MGTX
Sep 1 13:43:14	1.5735	1.5740	BBIX

Our goal is to model the time series and to understand individual differences among the banks. The data we have at our disposal consists of the tick data in Figure 3 together with various properties of the banks, such as their geographical location.

We hypothesize that the tick data is effectively a random walk, but where the percentage change at each time point is influenced by the bank posting the price. For example, we might suspect that some banks tend to post larger differences from the previous tick than the average change, or that some banks post more frequently during upswings than downswings, so that the ticks posted by such a bank run contrary to the downward trend. Figure 4 shows the kind of thing we are after, plotting the empirical frequency of percentage changes for all the ticks, for a bank that only posts large changes from the previous tick, and for a large bank that posts many changes.

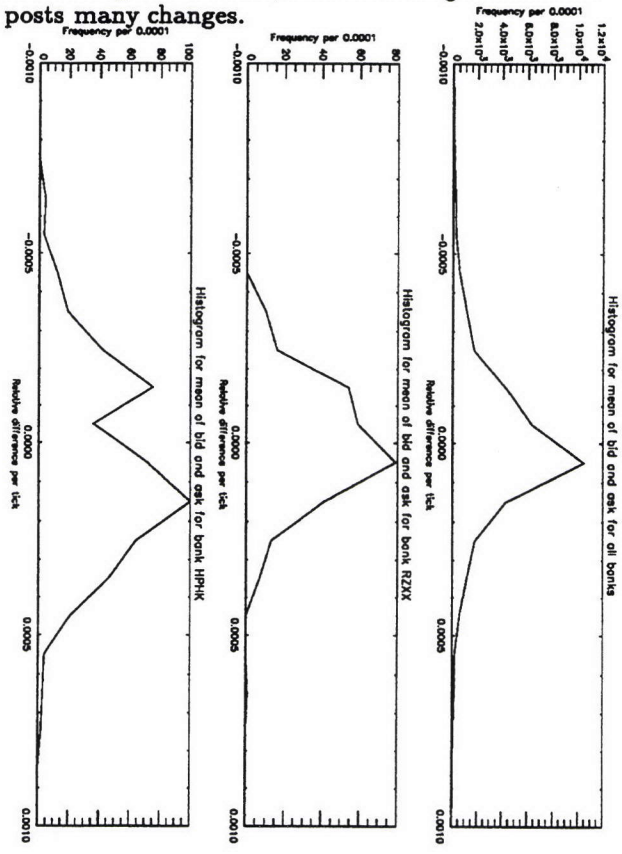

Figure 4: Frequencies for different relative price changes.

We sit down at our data analysis system, pull in the raw data, and set up a quick model to do an unsupervised clustering of the banks. Our first pass uses the random so that we can get a basic feel for the different kinds of banks. For each bank we have:

- The mean of the bank's bid-ask spread.
- The bank's geographical location.
- The average number of posts the bank generates per day.
- The massaged tick data giving the bank's relative price change over the immediately preceding price (probably posted by a different bank).

The graphical model, shown in Figure 5, is created using a drawing tool. In this model, the relative

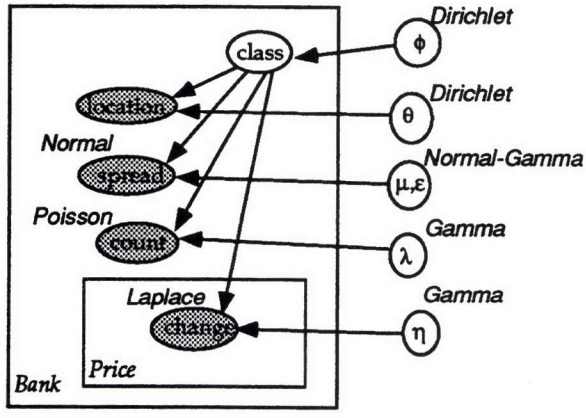

Figure 5: Basic clustered random work model for price changes.

change that a bank will offer is assumed to be determined by its class, but is otherwise a random walk. Notice this model has multiple banks, and each bank gives multiple prices, so this model has multiple plates indicated a double product occurs in the full joint distribution for the data (over banks and over the bank's price changes). For the current problem we simply drag a prefabricated mixture model and appropriate components from a palette or component library into the work area, and make some modifications to it. The model should include complete specification of all distributions and parameters (e.g., all parameters of the Dirichlets in Figure 5 be supplied). For instance, we would have to set the various parameters for the conjugate priors appearing in the model. The model of Figure 5 is not a standard clustering model so could not be obtained from any standard statistical package. In other problems we could create a free form model by drawing individual nodes, links, and probability annotations. The drawing tool contains the necessary hooks to associate variables on the plate with the fields from the bank database, and the computed fields from the tick database.

We now press the RUN button. At this point the system performs all the drudgery such as mathematical calculations, programming, and validation, previously requiring weeks of effort:

- It performs known symbolic simplifications on the graphical model. For instance, it knows about sufficient statistics and some closed form solutions to expected values.
- It computes all required derivative functions.
- It chooses an optimization algorithm.
- It finds the parameter values of maximum posterior probability, together with the Bayes factor and the Hessian of the posterior evaluated at the final parameters.
- It generates optimized C code to evaluate the model by performing a data flow analysis over the needed computations.

Now we could provide the system with an algorithm scheme, such as EM or Gibbs sampling (additional examples are given later), and have the system come up with the necessary code for the derivatives, expected values, probabilities, and so forth. However, in this case the default algorithm matching the graph is good enough.

We can now analyze the final model to see what it tells us, for instance using available visualization tools. (All this is make believe.) The classes that it finds are natural ones we might expect. The banks are broken into different classes according to whether they are closer to the New York or London markets. Banks that post infrequently tend to have a higher bid ask spread than those that post often. Each of these groupings also has different random walks for their pricing. Some smaller banks, for instance, tend to make larger changes.

We now go back to the drawing tool and refine the model to account for the context of the tick data. In the previous model, the relative change that a bank will offer is assumed to be determined by its class, but is otherwise a random walk. In this case we also make the price sensitive to the current average trend which is dependent on previous prices. This new model is given in Figure 6. Having drawn the new model, we simply click the RUN button and let the system re-optimize for the new configuration. This time, we could use the previous classification got as the initial values for the new extended model.

For this more complex model, we would probably have to modify the default algorithm scheme suggested by the system. This is something we expect in general, so the system we propose includes both a graphical model and a general algorithm scheme as inputs. If the algorithm scheme is missing, the system can provide a default using general purpose algorithms such as Gibbs, EM or MAP algorithms.

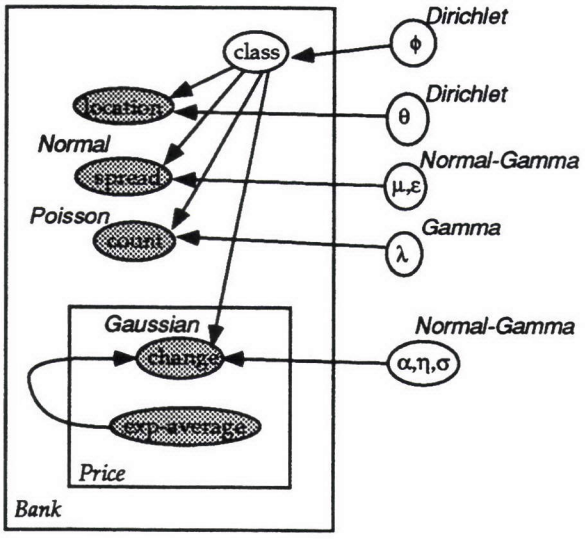

Figure 6: Trend sensitive model for price changes.

A more detailed example

An example of a graphical model for a simple clustering problem is given in Figure 7. To be explicit,

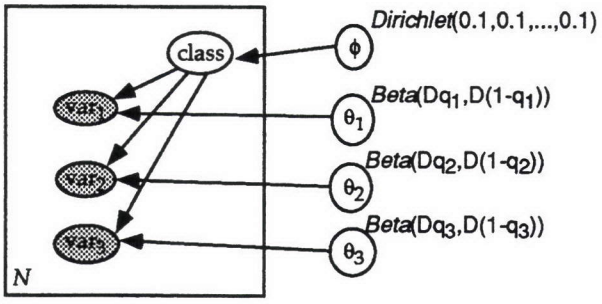

Figure 7: A simple unsupervised learning problem.

we also have to give the full distributions for the variables in the graph. Assume there are 10 classes, so $class \in \{1, 2, \ldots, 10\}$, and the variables var_i are binary. The graphical model is a mnenomic for the following distributional assumptions for the j-th case being

$$\begin{aligned} var_{i,j} &\sim \text{Bernoulli with prob. success } \theta_{i,class_j}; \\ class_j &\sim \text{10-dimensional Multinomial with} \\ &\qquad \text{probabilities } \phi_1, \phi_2, \ldots, \phi_{10}; \end{aligned}$$

and for the parameters ϕ and θ being

$$\begin{aligned} \phi &\sim Dirichlet(0.1, 0.1, \ldots, 0.1)\ ; \\ \theta_{i,c} &\sim Beta(Dq_i, D(1-q_i))\ ; \end{aligned}$$

for the hyper-parameters D, q_1, q_2, q_3. Using an empirical Bayes approach, we set q_i to be the observed frequency of success for var_i and set $D = 4$ to yield reasonable prior standard deviation for the possible values of θ_i.

A matching algorithm scheme for this model, an iterated EM algorithm in pseudo-code, goes as follows:

1. Repeat, 5 times.
   (a) Initialize the parameters ϕ, θ randomly according to their prior.
   (b) Repeat until the maximum relative difference in parameter values ϕ, θ is less than $1.0e-5$.
      i. Reassign the sufficient statistics for ϕ as follows:

$$ss(\phi) = \sum_{i=1}^{N} \mathcal{E}_{class_i|var_{1,i},var_{2,i},var_{3,i},\phi,\theta}(class)$$

      ii. For $c = 1, \ldots, 10$, reassign the sufficient statistics for $\theta_{1,c}, \theta_{2,c}, \theta_{3,c}$ as follows:

$$ss(\theta_{j,c}) = \sum_{i=1}^{N} \mathcal{E}_{class_i|var_{1,i},var_{2,i},var_{3,i},\phi,\theta}(1_{class_i=c} var_{j,i})$$

      iii. Replace ϕ and θ by their MAP values given the sufficient statistics as above.
   (c) Compute the score for the final parameter values as

$$score = \det\left(\frac{\mathrm{d}^2 \log p(\phi, \theta|sample)}{\mathrm{d}(\phi, \theta)}\right)$$

2. Return the parameters ϕ, θ matching the best score.

The system would automatically compute the derivatives, expected values, MAP calculations, and so forth and insert the code efficiently into the major loops. Notice there are problems with this algorithm scheme, for instance, if a ϕ_i approaches zero the score will become ill-defined because some of the parameters will become redundant. This is irrelevant for the purposes of our illustration.

High-level specification of data analysis algorithms

An overview of the basic framework is given in Figure 8. Inputs to the system are a graphical model to specify the variables and their relationships, and an algorithm scheme to specify the algorithm. These two inputs can be patched together from libraries.

A language to specify variables and their relationships

Probabilistic graphical models (chain graphs [44, 16]) extended with plates are used here as a specification language. When augmented with specific functional forms such as the Gaussian and the logistic, this language is sufficient powerful to represent a broad range of problems across several fields: generalized linear models, feed-forward networks, Jordan and Jacobs mixture of experts [27], unsupervised learning of many different kinds, and hybrids of these models. A review of some of the learning problems represented appears in [7].

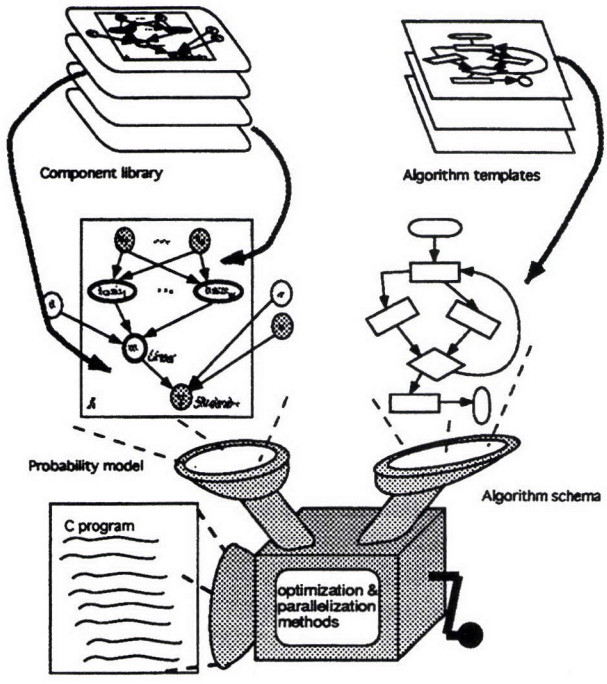

Figure 8: The basic framework for specification.

A graphical model implies a probability model, thus it defines how probabilities, log-probabilities, their derivatives, and some expected values can be computed, and in some cases how sampling can be done. For a Gaussian or discrete Bayesian network, for instance, the usual exact computations are implied by the graph [39]. Techniques for computing these quantities are of course more complex in other cases. In general, exact methods for expected values are not known, and probabilities are not in general easy to compute for chain graphs and Markov networks, although ratios of probabilities are. The automatic calculation of derivatives on structures such as graphs is a well understood problem [20]. In neural networks, this corresponds to the Backpropagation algorithm and its extensions for second derivatives [9]. Likewise, the calculation of derivatives on probabilistic graphical models is an application of the chain rule for differentiation. Details appear in [3].

Algorithm schemes

Algorithm schemes are high-level algorithms used as input for code generation and compilation. The algorithm scheme refers to variables, probabilities, log-probabilities, derivatives, and expected values for items on the graph whose computation can be determined automatically. Thus the scheme is free of many of the cumbersome equational details found in most languages. Algorithm schemes themselves can be produced automatically from algorithm templates for problems matching the right preconditions, and in other cases may be provided or refined by the user. We do not give a specification of the scheme and template language itself here, but the reader can infer from the examples that the scheme language is a fairly standard procedural language with appropriate hooks into the matching graphical model. Some sources for algorithm templates are as follows:

- Gilks *et al* [19] have developed general algorithms to perform Gibbs sampling on Bayesian networks with plates.
- Other algorithms such as conjugate gradient, Fisher's scoring method, or Laplace approximations [43, 28] can be applied once first and second derivatives are calculated for model parameters.
- Lauritzen describes the application of the EM algorithm [15] to Bayesian networks with a single plate [29] in the context of missing values. The more general application of the EM algorithm for hidden variables is obvious, as for instance done in unsupervised learning [21].

General discussion

We can see that many parts of this ambitious plan, a software tool kit for data analysis, are already in place. But the aim here is to provide tools for development, not complete packaged solutions. Proponents of Gibbs sampling, for instance, say that the design of an efficient sampler takes care and experience. For instance, specific matrix forms might be used to advantage. It is often the case that some fine tuning is needed in algorithms.

One task that can never have direct software support is the design of an appropriatee model with an appropriate prior. This is a knowledge elicitation problem. Techniques here are varied and range from careful choice of the representation to simplify elicitation [22], to techniques for working with components and libraries [1]. But the elicitation task still has to be done afresh with each different problem, except in those prototypical situations that are routinely addressed by standard statistical packages. While one might use a standard package in initial modeling, as the problem becomes better understood specific requirements are needed that canned software may not provide. Of course, tools for software generation alleviate the modeling task greatly by providing rapid prototyping. Nevertheless, it is my view that a sizeable burden in the Bayesian analysis of data is software engineering rather than the statistical analysis itself, and therefore software generators and support tools are both a realistic and important goal.

Naturally, an important part of such a framework is component libraries containing modules for common sub-tasks. Almond *et al.* [1] point out that parts of a graph, *components*, are often shared in a series of applications. Learning and data analysis are no different. One useful component is the generalized linear model [33] which can include basis function sets for orthogonal polynomials or wavelets. Likewise, algorithm templates for standard techniques such as Gibbs sampling and the EM algorithm could be prepared.

In the remainder of this paper we discuss a few more pieces for this general software toolkit. The first is an algorithm for the decomposition of a chain graph with plates into its independent components. This technique has been used to develop efficient algorithms for learning Bayesian networks from complete data [6, 11, 32]. The second contribution is some exact algorithms on graphical models with a single plate. Both these simplify calculation of the Bayes factor for a model, used widely in Bayesian methods [28, 32]. The Bayes factor is the support given to model M_2 relative to model M_1 by the data *sample*.

$$\mathit{Bayes\text{-}factor}(M_2, M_1) = \frac{p(\mathit{sample}|M_2)}{p(\mathit{sample}|M_1)} .$$

We use the term *evidence* [31] for the basic component, $\mathit{evidence}(M) = p(\mathit{sample}|M)$ and consider its calculation throughout.

While these techniques can be used in many places in a learning toolkit, one interesting by-product is that they show how to develop algorithms for learning DAGs from complete data where the conditional distributions are in the exponential family, including mixtures of Gaussians, Poissons, discrete variables, etc. All that is required is a conjugate prior. While this capability should not be surprising —and perhaps the hardest part, appropriate priors, is left out—it is interesting that we can construct these algorithms automatically using the operations presented here.

Exact algorithms on graphs with plates

The removal of a plate by arc reversal is related to arc reversal used for influence diagrams [38], however, the operation must be recursive so requires much stricter conditions to apply. These conditions are well known in statistics as the problem reduces

to the existence of sufficient statistics and applies to members of the exponential family of distributions. This includes standard undergraduate distributions such as Gaussians, Chi squared, and Gamma, and many more complex distributions constructed from simple components including class probability trees over discrete input domains [5], simple discrete and Gaussian versions of a Bayesian network [45], and linear regression with a Gaussian error. Thus, these are a broad and not insignificant class of distributions that are given in the definition below. Their general form has a linear combination of parameters and data in the exponential.

Definition 1 *A space X is* independent *of the parameter θ if the space remains the same when just θ is changed. If the domains of x, y are independent of θ, then the conditional distribution for x given y, $p(x|y,\theta,M)$, is in the* exponential family *when*

$$p(x|y,\theta,M) = \frac{h(x,y)}{Z(\theta)} \exp\left(\sum_{i=1}^{k} w_i(\theta) t_i(x,y)\right) , \quad (1)$$

for some functions w_i, t_i, h and Z and some integer k, for $h(x,y) > 0$. The normalization constant *$Z(\theta)$ is known as the* partition function.

The following is a simple graphical reinterpretation of the Pitman-Koopman Theorem from statistics [25, 14]. Consider Figure 9(a). $T(x_*, y_*)$ is a statistic of fixed dimension independent of the sample size N (corresponding to n, p in the coin tossing example). The theorem says that the sample in Figure 9(a) can be summarized in statistics, as shown in Figure 9(b), if and only if the probability distribution for $x|y,\theta$ is in the exponential family. In this case it is said that $T(x_*, y_*)$ is a sufficient statistic.

Theorem 1 *(*Recursive arc-reversal*). Consider the model M represented by the graphical model for a sample of size N given in Figure 9(a). Have x in the domain X and y in the domain Y, both domains are independent of θ, and both domains have components that are real valued or finite discrete. Let the conditional distribution for x given y,θ be $f(x|y,\theta)$, which is positive for all $x \in X$. If first derivatives exist w.r.t. all real valued components of x and y, the plate removal operation applies for all samples $x_* = x_1, \ldots, x_N$, $y_* = y_1, \ldots, y_N$, and θ, as given in Figure 9(b), for some sufficient statistics $T(x_*, y_*)$ of dimension independent of N if and only if the conditional distribution for x given y,θ is in the exponential family, given by Equation (1). In this case, $T(x_*, y_*)$ is an invertible function of the k averages*

$$\frac{1}{N}\sum_{j=1}^{N} t_i(x_j) \quad : \quad i = 1, \ldots, k .$$

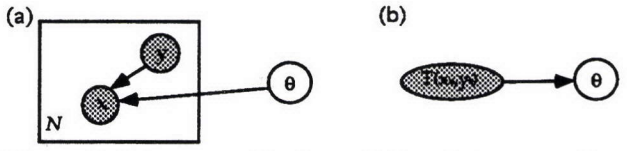

Figure 9: The generalized graph for plate removal

Graph decomposition

Learning problems can be decomposed into subproblems in some cases. For instance, consider the learning problem given in Figure 10 over two multinomial variables var_1 and var_2, and two Gaussian variables x_1 and x_2. For this problem we have specified two alternative models, model M_1 and model M_2. Model M_2 has an additional arc going from the discrete variable var_2 to the real valued variable x_1. We will use this subsequently to discuss local search of these models evaluated by their Bayes factor.

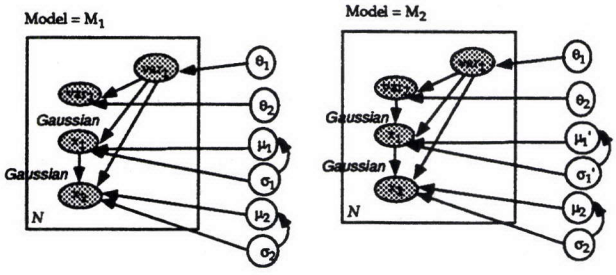

Figure 10: Two graphical models

A straight forward manipulation of the conditional distribution for this model yields, for model M_1, the conditional distribution given in Figure 11. When parameters, θ_1, θ_2, etc., are *a priori* independent, and their data likelihoods do not introduce cross terms between them, the parameters become *a posteriori* independent as well. This occurs for θ_1, θ_2, and the set $\{\mu_1, \sigma_1\}$. This model simplification also implies the evidence for model M_1 decomposes similarly. Denote the sample of the variable

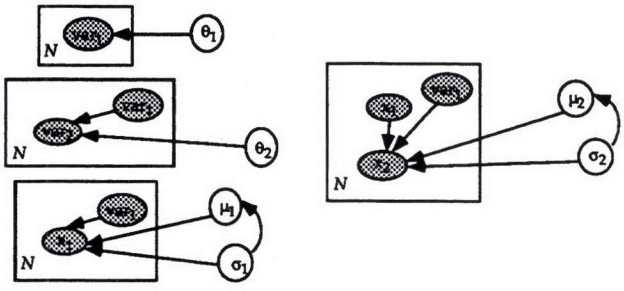

Figure 11: A simplification of model M_1

x_1 as $x_{1,*} = x_{1,1}, \ldots, x_{1,N}$, and likewise for var_1 and var_2, etc. In this case, we get,

$$evidence(M_1) = p(var_{1,*}|M_1)\, p(var_{2,*}|var_{1,*}, M_1) \quad (2)$$
$$p(x_{1,*}|var_{1,*}, M_1)\, p(x_{2,*}|x_{1,*}, var_{1,*}, M_1) .$$

The evidence for model M_2 is similar except that the posterior distribution of μ_1 and σ_1 is replaced by the posterior distribution for μ_1' and σ_1'.

This result is general, and applies to both DAGs, undirected graphs, and more generally to chain graphs. Similar results results are covered by Dawid and Lauritzen [12] for a family of models they call hyper-Markov. The general result described above is an application of the rules of independence applied to plates. This uses a notion of local dependence, which is called the Markov blanket, following Pearl [35]. The Markov blanket is a nodes parents, children, and the children's parents. If deterministic nodes are involved, the definition requires a bit more care [3].

To perform the simplification depicted in Figure 11, it is sufficient then to find the finest partitioning of the model parameters such that they are independent. The decomposition in Figure 11 represents the finest such partition of model M_1. The evidence for the model will then factor according to the partition, as given for model M_1 in Equation (2). For this task we have the following theorem.

Theorem 2 *(Decomposition). A model M is represented by a chain graph G with plates and no deterministic nodes. Let the variables in the graph be X. We have P possibly empty subsets of the variables X, X_i for $i = 1, \ldots, P$ such that $unknown(X_i)$ is a partition of $unknown(X)$. This induces a decomposition of the graph G into P subgraphs G_i where:*

- *the graph G_i contains the nodes X_i and any arcs and plates occuring on these nodes; and*
- *the potential functions for cliques in G_i are equivalent to those in G.*

The induced decomposition represents the unique finest equivalent independence model to the original graph if and only if X_i for $i = 1, \ldots, P$ is the finest collection of sets such that, when ignoring plates, for every unknown node u in X_i, its Markov blanket is also in X_i. This finest decomposition takes $O(|X|^2)$ to compute. Furthermore, the evidence for M now becomes a product over each subgraph,

$$evidence(M) = f_0 \prod_i f_i(known(X_{i,*})) , \quad (3)$$

for some functions f_i (given in the proof).

In some cases, the functions f_i have a clean interpretation: they are equal to the evidence for the subgraphs. This result can be obtained from the following corollary.

Corollary 2.1 *In the context of Theorem 2 where there are no deterministic nodes, suppose there exists a set of chain components τ_j from the graph ignoring plates such that $X_j = \tau_j \cup parents(\tau_j)$, where $unknown(parents(\tau_j)) = \emptyset$. Then*

$$f_j(known(X_{j,*})) = p(known(\tau_j)_*|parents(\tau_j)_*, M) .$$

If we denote the j-th subgraph by model M_j, then this term is the conditional evidence for model M_j given $parents(\tau_j)_*$. Denote by M_0 the subgraph on known variables induced by $cliques_0$ (as given in the proof). If the condition of Corollary 2.1 holds for M_j for $j = 0, 1, \ldots, P$, then it follows that the evidence for the model M is equal to the product of the evidence for each subgraph.

$$evidence(M) = \prod_{i=0}^{P} evidence(M_i) . \quad (4)$$

This holds in general if the original graph G is a DAG, as used in learning DAGs [6, 11].

Corollary 2.2 *Equation (4) holds if the parent graph G is a DAG with plates.*

In general, we might consider searching through a family of graphical models. To do this we can use local search [26] or numerical optimization to find high posterior models, or Markov chain Monte Carlo methods [34] to select a sample of representative models [3]. To do this, we first show how to represent a family of models. Figure 12, for instance, is similar to models of Figure 10 except that some arcs are hatched. We use this to indicate that

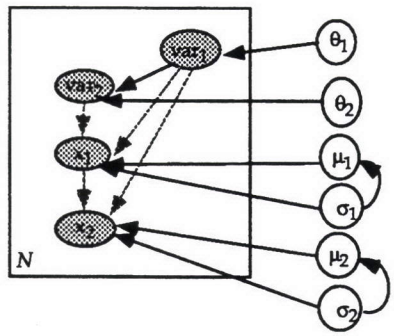

Figure 12: A family of models (optional arcs hatched)

these arcs are optional. To instantiate a hatched arc they can either be removed, or replaced with a full arc. This graphical model then represents many different models, for all 2^4 possible instantiations of the arcs. Prior probabilities for these models could be generated using a scheme such as in [6, p54] where a prior probability is assigned by a domain expert for the inclusion of each arc, and

the prior for a full model found by multiplication. The family of models given by Figure 12 includes those of Figure 10 as instances. During search or sampling, an important property is the Bayes factor for the two models, $Bayes\text{-}factor(M_2, M_1)$. Because of the decompositions above, the Bayes factor can be found by only examining component Bayes factors for nodes whose parents have changed between models M_1 and M_2. The difference here is the model for the variable x_1.

$$Bayes\text{-}factor(M_2, M_1) = \frac{p(x_{1,*}|var_{1,*}, var_{2,*}, M_2)}{p(x_{1,*}|var_{1,*}, M_1)}$$

That is, the Bayes factor can be computed from only considering the models involving μ_1, σ_1 and μ_1', σ_1'.

This incremental modification of evidence, Bayes factors, and finest decompositions is also general, and follows directly from the independence test. It has been used in fast learning algorithms for DAGs [5, 6, 11], and a similar property for undirected graphs is given in [12]. This is developed below for the case of directed arcs and non-deterministic variables.

Lemma 1 *For a graph G in the context of Theorem 2 with no deterministic nodes, we have two variables U and V such that U is given. Consider adding/removing a directed arc from U to V. We update the finest decomposition of G as follows: There is a unique subgraph containing the unknown variables in parents(chain-component(V)). To this subgraph add/delete an arc from U to V, and add/delete U to the subgraph if required.*

We can therefore add shaded non-deterministic parents at will to nodes in a graph and the finest decomposition remains unchanged except for a few additional arcs. The use of hatched arcs in these contexts therefore causes no additional trouble to the decomposition process. That is, we form the finest decomposition for a graph with plates and hatched directed arcs as if the arcs where normal directed arcs, and the evidence is adjusted during the search by adding the different parents as required.

Bayes factors for the exponential family

The above results are useful, but to make use of them automatically we need to be able to generate Bayes factors or evidence for models. It generally holds that if a likelihood is in the exponential family, then the posterior distribution for the model parameters is also in the exponential family, although it is only really useful when the normalizing constant is readily computed. This holds for the Dirichlet, the conjugate to a multinomial, and the Gaussian-Wishart, the conjugate to a Gaussian [14]. We give the results here.

Let the normalizing constant for the distribution of Theorem 1 be $Z_1(\theta)Z_2$ where Z_2 is a constant part independent of θ. Assume the prior on θ takes a conjugate form such as:

$$p(\theta|\tau, M) = \frac{f(\theta)}{Z_\theta(\tau)} \exp\left(\tau_{k+1}(\log 1/Z_1(\theta)) + \sum_{i=1}^{k} \tau_i w_i(\theta)\right), \quad (5)$$

for some $k+1$ dimensional parameter τ, where $Z_\theta(\tau)$ is the appropriate normalizing constant and $f(\theta)$ is any function. Then the posterior distribution is also conjugate and the Bayes factor can be readily computed.

Lemma 2 *Consider the context above. Then the* model likelihood *or* evidence, *given by evidence(M) = $p(x_1, \ldots, x_N|y_1, \ldots, y_N, M)$, can be computed as:*

$$\begin{aligned} evidence(M) &= \frac{p(\theta|\tau)\prod_{j=1}^{N} p(x_j|y_j, \theta)}{p(\theta|\tau')} \\ &= \frac{Z_\theta(\tau')}{Z_\theta(\tau)Z_2^N}. \end{aligned}$$

Learning mixed Bayesian networks

Class probability trees and discrete Bayesian networks can be learned efficiently by noticing that their basic form is exponential family [5, 4, 6, 11, 40]. Take, for instance, the family of models specified by the Bayesian network given in Figure 12. In this case, the Local Evidence Corollary, Corollary 2.1, applies. The evidence for Bayesian networks generated from this graph is therefore a product over the nodes in the Bayesian network. If we change a Bayesian network by adding or removing an arc, the Bayes factor is therefore simply the local Bayes factor for the node, as mentioned in the Incremental Decomposition Lemma, Lemma 1. Local search is then quite fast, and Gibbs sampling over the space of Bayesian networks is possible. A similar situation exists with trees [5]. The same results apply to any Bayesian network with exponential family distributions at each node, such as Gaussians, or a Poisson. Results for Gaussians and mixed discrete and Gaussian networks are presented, for instance, in [17, 3].

This local search approach is a MAP approach since it searches for the network structure maximizing posterior probability. We can do a more accurate approximation and generate a Markov chain of Bayesian networks from the search space of Bayesian networks. Because the Bayes factors are readily computed in this case, we can do Gibbs sampling or one of many other Markov chain Monte Carlo schemes. The scheme given below is the

Metropolis' algorithm [37], suggested because it only looks at single neighbors until a successor is found. This is done be repeating the following steps:

1. For the initial Bayesian network G, randomly select a neighboring Bayesian network G' differing only by an arc.
2. Compute $\textit{Bayes-factor}(G', G)$ by making the decompositions described in Theorem 2, and so doing a local computation as described in Lemma 1, and using the Bayes factors computed with Lemma 2.
3. Accept the new Bayesian network G' with probability given by

$$\min\left(1, \textit{Bayes-factor}(G', G)\frac{p(G')}{p(G)}\right) .$$

   If accepted, assign G' to G, otherwise G remains unchanged.

A local maxima Bayesian network could of course be found concurrently, however, this scheme generates a set of Bayesian networks appropriate for model averaging and expert evaluation of the space of potential Bayesian networks. Of course, initialization might search for local maxima to use as a reference.

References

[1] R.G. Almond, J.M. Bradshaw, and D. Madigan. Reuse and sharing of graphical belief network components. In Cheeseman and Oldford [10], pages 113–122.

[2] G.L. Bretthorst. An introduction to model selection using probability theory as logic. In G. Heidbreder, editor, *Maximum Entropy and Bayesian Methods*. Kluwer Academic, 1994. Proceedings, at Santa Barbara, 1993.

[3] W. Buntine. Operations for learning with graphical models. *Journal of Artificial Intelligence Research*, 1994. to appear.

[4] W.L. Buntine. Classifiers: A theoretical and empirical study. In IJCAI91 [24].

[5] W.L. Buntine. Learning classification trees. In D.J. Hand, editor, *Artificial Intelligence Frontiers in Statistics*, pages 182–201. Chapman & Hall, London, 1991.

[6] W.L. Buntine. Theory refinement of Bayesian networks. In B.D. D'Ambrosio, P. Smets, and P.P. Bonissone, editors, *Uncertainty in Artificial Intelligence: Proceedings of the Seventh Conference*, Los Angeles, CA, 1991.

[7] W.L. Buntine. Representing learning with graphical models. Technical Report FIA-94-14, Artificial Intelligence Research Branch, NASA Ames Research Center, 1994. Submitted.

[8] W.L. Buntine and A.S. Weigend. Bayesian back-propagation. *Complex Systems*, 5(1):603–643, 1991.

[9] W.L. Buntine and A.S. Weigend. Computing second derivatives in feed-forward networks: a review. *IEEE Transactions on Neural Networks*, 5(3), 1994.

[10] P. Cheeseman and R.W. Oldford, editors. *Selecting Models from Data: Artificial Intelligence and Statistics IV*. Springer-Verlag, 1994.

[11] G.F. Cooper and E.H. Herskovits. A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9(4):309–348, 1992.

[12] A.P. Dawid and S.L. Lauritzen. Hyper Markov laws in the statistical analysis of decomposable graphical models. *Annals of Statistics*, 21(3):1272–1317, 1993.

[13] R. Lopez de Mantaras and D. Poole, editors. *Uncertainty in Artificial Intelligence: Proceedings of the Tenth Conference*, Seattle, WA, 1994.

[14] M.H. DeGroot. *Optimal Statistical Decisions*. McGraw-Hill, 1970.

[15] A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society B*, 39:1–38, 1977.

[16] M. Frydenberg. The chain graph Markov property. *Scandinavian Journal of Statistics*, 1991.

[17] D. Geiger and D. Heckerman. Learning Gaussian networks. In de Mantaras and Poole [13], pages 235–243.

[18] W.R. Gilks, D.G. Clayton, D.J. Spiegelhalter, N.G. Best, A.J. McNeil, L.D. Sharples, and A.J. Kirby. Modelling complexity: applications of Gibbs sampling in medicine. *Journal of the Royal Statistical Society B*, 55:39–102, 1993.

[19] W.R. Gilks, A. Thomas, and D.J. Spiegelhalter. A language and program for complex Bayesian modelling. *The Statistician*, 1993.

[20] Andreas Griewank and George F. Corliss, editors. *Automatic Differentiation of Algorithms: Theory, Implementation, and Application*, Breckenridge, Colorado, January 6–8 1991. SIAM.

[21] R. Hanson, J. Stutz, and P. Cheeseman. Bayesian classification with correlation and inheritance. In IJCAI91 [24].

[22] David Heckerman. *Probabilistic Similarity Networks*. MIT Press, 1991.

[23] J.A. Hertz, A.S. Krogh, and R.G. Palmer. *Introduction to the Theory of Neural Computation*. Addison-Wesley, 1991.

[24] IJCAI91, editor. *International Joint Conference on Artificial Intelligence*, Sydney, 1991. Morgan Kaufmann.

[25] H. Jeffreys. *Theory of Probability*. Clarendon Press, Oxford, third edition, 1961.

[26] D.S. Johnson, C.H. Papdimitriou, and M. Yannakakis. How easy is local search? In *FOCS'85*, pages 39–42, 1985.

[27] M.I. Jordan and R.I. Jacobs. Supervised learning and divide-and-conquer: A statistical approach. In *Machine Learning: Proc. of the Tenth International Conference*, pages 159–166, Amherst, Massachusetts, 1993. Morgan Kaufmann.

[28] R.E. Kass and A.E. Raftery. Bayes factors and model uncertainty. Technical Report #571, Department of Statistics, Carnegie Mellon University, PA, 1993. Submitted to Jnl. of American Statistical Association.

[29] S.L. Lauritzen. The EM algorithm for graphical association models with missing data. *Computational Statistics and Data Analysis*, 1993. To appear; available as Tech. Rep. R 91-05, Institute for Electronic Systems, Aalborg University.

[30] D.J.C. MacKay. A practical Bayesian framework for backprop networks. *Neural Computation*, 4:448–472, 1992.

[31] D.J.C. MacKay. Bayesian non-linear modeling for the energy prediction competition. Report Draft 1.2, Cavendish Laboratory, University of Cambridge, 1993.

[32] D. Madigan, A.E. Raftery, J.C. York, J.M. Bradshaw, and R.G. Almond. Strategies for graphical model selection. In Cheeseman and Oldford [10], pages 91–100.

[33] P. McCullagh and J.A. Nelder. *Generalized Linear Models*. Chapman and Hall, London, second edition, 1989.

[34] R.M. Neal. Probabilistic inference using Markov chain Monte Carlo methods. Technical Report CRG-TR-93-1, Dept. of Computer Science, University of Toronto, 1993.

[35] J. Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, 1988.

[36] S.J. Press. *Bayesian Statistics*. Wiley, New York, 1989.

[37] B.D. Ripley. *Stochastic Simulation*. John Wiley & Sons, 1987.

[38] R.D. Shachter. Evaluating influence diagrams. *Operations Research*, 34(6):871–882, 1986.

[39] R.D. Shachter, S.K. Andersen, and P. Szolovits. Global conditioning for probabilistic inference in belief networks. In de Mantaras and Poole [13], pages 514–522.

[40] D.J. Spiegelhalter, A.P. Dawid, S.L. Lauritzen, and R.G. Cowell. Bayesian analysis in expert systems. *Statistical Science*, 8(3):219–283, 1993.

[41] D.J. Spiegelhalter and S.L. Lauritzen. Sequential updating of conditional probabilities on directed graphical structures. *Networks*, 20:579–605, 1990.

[42] M.A. Tanner. *Tools for Statistical Inference*. Springer-Verlag, New York, second edition, 1993.

[43] L. Tierney and J.B. Kadane. Accurate approximations for posterior moments and marginal densities. *Journal of the American Statistical Association*, 81(393):82–86, 1986.

[44] N. Wermuth and S.L. Lauritzen. On substantive research hypotheses, conditional independence graphs and graphical chain models. *Journal of the Royal Statistical Society B*, 51(3), 1989.

[45] J. Whittaker. *Graphical Models in Applied Multivariate Statistics*. Wiley, 1990.