4 Graphical Models for Discovering Knowledge

Wray Buntine
Research Institute for Advanced Computing Sciences, Computational Sciences Division, NASA Ames Research Center

Abstract

There are many different ways of representing knowledge, and for each of these ways there are many different discovery algorithms. How can we compare different representations? How can we mix, match and merge representations and algorithms on new problems with their own unique requirements? This chapter introduces probabilistic modeling as a philosophy for addressing these questions and presents graphical models for representing probabilistic models. Probabilistic graphical models are a unified qualitative and quantitative framework for representing and reasoning with probabilities and independencies.

4.1 Introduction

Perhaps one common element of the discovery systems described in this and previous books on knowledge discovery is that they are all different. Since the class of discovery problems is a challenging one, we cannot write a single program to address all of knowledge discovery. The KEFIR discovery system applied to health care by Matheus, Piatetsky-Shapiro, and McNeill (1995), for instance, is carefully tailored for a particular class of situations and could not have been easily used on the SKICAT application (Fayyad, Djorgovski, and Weir 1995). I do not know of a universal learning or discovery algorithm (Buntine 1990), and a universal problem description for discovery is arguably too broad to be used as a program specification.

As a consequence, the power to perform in an application lies in the way knowledge about the application is obtained, used, represented and modified. Unfortunately with today's technology, it is not possible to dump data into a discovery system and later read off the dollar savings. Rather one has to work closely with the experts involved, for instance in selecting and customizing tools. See the chapter by Brachman and Anand (1995) in this book for an account of the interactive aspects of knowledge discovery.

It is important then to have knowledge discovery techniques that allow flexibility in the way knowledge can be encoded, represented and discovered. Probabilistic graphical models offer such a technique. Probabilistic graphical models are a framework for structuring, representing and decomposing a problem using the notion of conditional independence. They have special cases and variations including Bayesian networks, influence diagrams, Markov networks, and causal probabilistic networks. These models are useful for the same reason that constraint satisfaction graphs are used in scheduling, data flow diagrams are used in scientific modeling, and fault trees are used in systems health management. They allow access to the structure of the problem without getting bogged down in the mathematical detail. Probabilistic graphical models do this by representing the variables in a problem and the relationships between them. Associated with graphical models themselves are the mathematical details such as the equations linking variables in the model, and algorithms for performing exact and approximate inference on the model.

Probabilistic graphical models are an attractive modeling tool for knowledge discovery because:

- They are a lucid representation for a variety of problems, allowing key dependencies within a problem to be expressed and irrelevancies to be ignored. They are flexible enough to represent supervised and unsupervised learning systems, neural networks, and many hybrids.
- They come with well understood techniques for key tasks in the discovery process:
  - problem formulation and decomposition,
  - designing a learning algorithm (Buntine 1994),
  - identification of valuable knowledge (using decision theory), and
  - generation of explanations (Madigan, Mosurski, and Almond 1995).

Only a simple form of graphical model is considered in this chapter, the Bayesian network. Reasoning about the value of knowledge on Bayesian networks can be done by adding "value" nodes, and using the tools of influence diagrams and utility theory (Shachter 1986), part of modern decision theory. This is not covered in this chapter. Bayesian networks are introduced in Section 4.2, problem decomposition is discussed in Section 4.3, knowledge refinement is discussed in Section 4.4, and relationships to a variety of learning representations are discussed in Section 4.5. Implications to discovery are given in the conclusion.

4.2 Introduction to graphical models

Graphs are used to represent *models*. A model in general is some proposed representation of the problem at hand showing the different variables involved, data and parameters, and the probabilistic or deterministic relationships between them. The basic model we consider consists of nodes representing variables, and arcs that indicate dependencies between variables (or, no arcs indicating independencies). The *variables* represented may be real valued or discrete, and may be

- variables whose values are given in the data,
- "hidden" variables believed to exist such as medical syndromes or hypothesized classes in a data base of stars, or
- *parameters* used to specify a model such as the weights in a neural network, the standard deviation of a Gaussian, the radius of diffusion in an instrument, or the error rate along a transmission channel.

These are all variables but often considered different from data. Their difference being that some might have their values currently known, some might be revealed to us in the future, some we might reasonably measure indirectly, and some we only hypothesize they exist and use the calculus of probability to estimate.

Below we introduce the basic kind of graphical model, a Bayesian network, and give a brief insight into its interpretation. This brief tour is necessary before applying graphical models to discovery and learning. A Bayesian network is a graphical model that uses directed arcs exclusively to form an directed acyclic graph (i.e., a directed graph without directed cycles). Figure 4.1, adapted from Shachter and Heckerman (1987), shows a

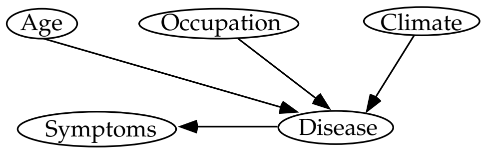

Figure 4.1
A simplified medical problem.

simple Bayesian network for a simplified medical problem. This graph represents a *domain model* for the problem. This organizes variables in the way the medical specialist would usually like to understand the problem, and arcs in the graph intuitively correspond to the notion *can cause* or *influence*. For instance, it may be thought that disease causes symptoms, and that age, occupation and climate causes disease. Under no stretch of the imagination could a disease be said to be caused by its symptoms[1]. The graph of Figure 4.1 can also be called a *causal model*. Other graphical models might represent the variables in a different ordering depending on whether the graph is being used to represent the domain model, a *computational model* for use by a program, or a particular view representative of some user. Graphical models can be manipulated to represent all these different views of a probabilistic knowledge base.

Graphical models are a language for expressing problem decomposition. They show how to decompose a problem into simpler subproblems. For a directed acyclic graph, this is done by a conditional decomposition of the joint probability (see, for instance, Lauritzen *et al.* [1990], and Pearl [1988] for more detail including other interpretations). This is as follows (full variable names have been abbreviated). M here represents the context. All probability statements are relative to context (context is dropped in later discussions for brevity).

$$\begin{aligned} p(Age, Occ, Clim, Dis, Symp|M) \; = \; & \qquad (4.2.1) \\ p(Age|M)\, p(Occ|M)\, p(Clim|M)\, p(Dis|Age, Occ, Clim, M)\, p(Symp|Dis, M) \; . & \end{aligned}$$

Each variable is written down conditioned on its *parents*, where $parents(x)$ is the set of variables with a directed arc into x. The general form for this for a set of variables X is

$$p(X|M) \; = \; \prod_{x \in X} p(x|parents(x), M) \; . \qquad (4.2.2)$$

Compare Equation (4.2.1) with one way of writing the *complete* joint probability:

$$\begin{aligned} p(Age, Occ, Clim, Dis, Symp|M) \; = \; & p(Age|M)\, p(Occ|Age, M) \qquad (4.2.3) \\ p(Clim|Age, Occ, M)\, p(Dis|Age, Occ, Clim, M)\, & p(Symp|Age, Occ, Clim, Dis, M) \; . \end{aligned}$$

This complete joint is an identity of probability theory, and makes no independence assumptions about the problem.

Probability models such as these are used primarily for performing *inference* on new problems. Graphical models are useful here because many kinds of inference can be performed on them. Basic inference involves calculating probabilities for arbitrary sets of variables (Shachter, Andersen and Szolovits 1994). Graphical models have been used in domains such as diagnosis, probabilistic expert systems, in planning and control (Dean

[1]Unless there was some kind of time delay and feedback involved.

and Wellman 1991; Chan and Shachter 1992), and in statistical analysis of data (Gilks, Thomas, and Spiegelhalter 1993), which is often more goal directed than typical knowledge discovery. Graphical models also generalize some aspects of Kalman filters (Poland 1994) used in control and hidden Markov models, the basic tool used in speech recognition (Rabiner and Juang 1986) and fault diagnosis (Smyth and Mellstrom 1992). Therefore graphical models are also used for dynamic systems and forecasting (Kjæruff 1992; Dagum *et al.* 1995). Various methods for learning simple kinds of graphical models from data also exist (Heckerman 1995). More extensive introductions to probabilistic graphical models can be found in (Henrion, Breese, and Horvitz 1991; Whittaker 1990; Pearl 1988; Spiegelhalter *et al.* 1993), and to learning in graphical models can be found in (Spiegelhalter *et al.* 1993; Buntine 1994; Heckerman 1995).

4.3 Problem decomposition

Learning and discovery problems rarely come neatly packaged and labeled according to their type. It is common for the practitioner to spend some time analyzing a problem as to how and where data analysis should be applied. This analysis and decomposition of a problem is routinely done for knowledge acquisition and software development, but has not attracted as much attention in the data analysis, discovery and learning literature. This section introduces the technique of problem decomposition using graphical models. The reasons for doing decomposition are two-fold. First and clearly, simplifying a problem is good in itself. Second and more importantly, a simpler model is easier to learn from data because it has less parameters. This makes discovery feasible and more reliable. Graphical models are a convenient way of making the structure of the decomposition apparent without going into the precise mathematical detail.

This section illustrates the process of problem decomposition by working through an example of topic spotting. Several other examples could equally well have illustrated this process. The topic spotting example addresses two common problems in supervised learning: a large input space and a multi-class decision problem.

Associated Press produces short newswires at a rate of tens of thousands per year. These come in approximately 90 broad topics and contain in all some 11,000 different words. Although a single newswire may only be 400 words long. A typical newswire is given below.

> PRECIOUS METALS CLIMATE IMPROVING, SAYS MONTAGU
> LONDON, April 1 – The climate for precious metals is improving with prices benefiting from renewed inflation fears and the switching of funds from dollar and stock markets ... Silver prices in March gained some 15 pct in dollar

terms due to a weak dollar and silver is felt to be fairly cheap relative to gold ... The report said the firmness in oil prices was likely to continue in the short term ...
REUTER

The topics for this newswire are gold, silver and precious metals. The topics for any given newswire are often given in the subject line, as written by the author of the newswire. However, we ignore this for the purposes of illustration.

Suppose we wish to predict the topics from the text of the newswire, ignoring the subject line. The naive approach is to attempt to predict the 90 topics from the 400 words using a monolithic classifier with 11,000 inputs. Instead, this problem can be readily decomposed: The 90 or so topics can be broken down into sub-topics and co-topics because the topic space has a rich structure. Moreover, the space of input words has structure itself: Suppose a newswire is known to have the topic "precious metals". The presence of the word "beef" is irrelevant when trying to determine whether the sub-topic is gold or silver. However, the word "beef" would be relevant if the topic were known to be relevant to agriculture.

A partial decomposition for this problem is given in Figure 4.2. These three Bayesian

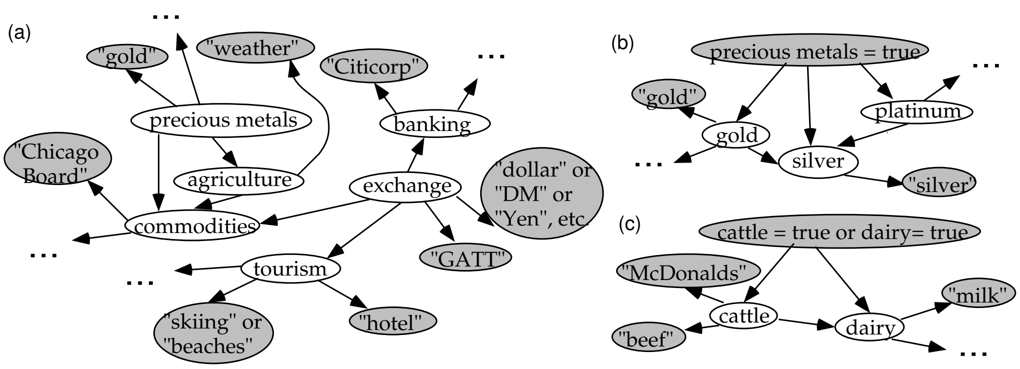

Figure 4.2
Three components of a topics-subtopics model (shaded nodes have known values).

networks are different to the previous in that some nodes are shaded and some are not. By convention, shaded nodes have their values known at the time of inference, and unshaded nodes do not. The partial decomposition goes as follows: First, we break the 90 topics up into groups. In Figure 4.2(a) these are the boolean variables *agriculture*, *precious-metals*, *tourism* and so forth. These topic variables can be recognized as the unshaded nodes in the graph. This graph is a model for these topics conditioned on the presence of various words in the newswire text. Variables consisting of quoted words indicate whether the word appears in the text. For instance, the variable *"gold"* in Figure 4.2(a) will be true if the word gold appears in the text, and false otherwise. Note this is different to whether the topic of the text is gold. In practice, word frequency counts are used and there are many more hundreds of words. Ignore this complication for the purposes of illustration. Also, all these quoted variables appear in shaded nodes. This indicates that we have the text before us, so we know the value of each of these word variables, whereas we do not know the topics.

Each topic now has its own graph to predict subtopics, and perhaps sub-subtopics. For instance, Figure 4.2(b) shows a sample subtopic graph for precious metals. Notice this graph has the top boolean variable *precious-metals* whose value is known to be true. This notation is used to indicate that this subgraph is contingent on *precious-metals* being a true topic. Likewise, Figure 4.2(c) shows a graph contingent on either *cattle* or *diary* being true. This graph assumes at least one of them is true and is used to predict whether one, the other, or both are true.

Probability is the unifying framework used to combine these different graphical models into a global model to predict the complete set of topics. This is done as follows: Adapting Equation (4.2.2) for the three graphs of Figure 4.2 yield three formulae for the following probabilities:

- $p(precious\text{-}metals, banking, exchange, commodities, agriculture, tourism|$ $\quad\quad$ $\text{``}ChicagoBoard\text{''}, \text{``}gold\text{''}, \text{``}weather\text{''}, \text{``}Citicorp\text{''}, etc.)$
- $p(gold, silver, platinum|precious\text{-}metals = true, \text{``}gold\text{''}, etc.)$
- $p(dairy, cattle|dairy = true\ OR\ cattle = true, \text{``}beef\text{''}, \text{``}McDonald's\text{''}, etc.)$

Likewise, corresponding formulae are obtained for the other graphs not depicted here. These probabilities can then be manipulated and combined to yield individual probabilities. For instance, suppose we wish to evaluate the probability $p(silver|newswire)$, where $newswire$ indicates the contents of the newswire is given and so all the words like $\text{``}beef\text{''}$ are also given. This can be computed using the two probability identities:

$$p(silver|newswire) = \\ p(silver|precious\text{-}metals = true, newswire)\, p(precious\text{-}metals = true|newswire)$$

$$p(silver|precious\text{-}metals = true, newswire) = \\ \sum_{gold \in \{T,F\}} \sum_{platinum \in \{T,F\}} p(gold, silver, platinum|precious\text{-}metals = true, newswire)$$

where $p(precious\text{-}metals = true|newswire)$ is computed similarly by summing out the other topic variables in Figure 4.2(a). Methods for combining probabilities from multiple networks can involve more complex schemes. A method developed for medical diagnosis that is suitable to the topic spotting problem considered here is similarity networks (Heckerman 1990). This is based on many graphs of the form of Figure 4.2(c) used to distinguish pairs of topics.

There are number of interesting questions for this decomposition approach. How do we develop such a decomposition? In diagnosis domains such as medicine, this kind of decomposition has been done manually in the development of probabilistic expert systems. It is found that experts are able to explain their own decompositions of a problem. Second, how can the decomposition be done automatically? While this is an open research question, standard techniques for learning should adapt to the task.

4.4 Knowledge Refinement

Unsupervised learning is a standard tool in statistics and pattern recognition. A well known example in discovery is the Autoclass application to the IRAS star database (Cheeseman and Stutz 1995). While these applications of unsupervised learning sometimes proceed routinely, it is more often the case that discovery is an iterative process. Initial exploration reveals some details and the discovery algorithm is modified as a result. Here, the discovery process parallels the iterative refinement strategies popular in software engineering. These strategies are made possible by rapid prototyping software such as Tcl/Tk used for developing interfaces (Ousterhout 1994). This aspect of discovery is discussed further by Brachman and Anand (1995). The application of iterative refinement to knowledge discovery and knowledge acquisition is one way of viewing knowledge refinement (Ginsberg, Weiss, and Politakis 1988; Towell, Shavlik, and Noordewier 1990).

An application where this kind of refinement was required is the analysis of aviation safety data given by Kraft and Buntine (1993). The task was to discover classes of aircraft incidents. In this case, standard unsupervised learning revealed incident classes that the domain expert believed were confounded by basic relationships expected in the data. A graphical model illustrating and simplifying the standard unsupervised learning is given in Figure 4.3. The algorithm used in this initial investigation was an algorithm called SNOB (Wallace and Boulton 1970), related to Autoclass. This algorithm builds a classification model as represented in the figure. For a given aircraft incident, details are recorded on the pilot, the controller, the kind of aircraft, its mission, and other information. Figure 4.3 indicates that if a set of aircraft is of the same hidden incident class, then the details recorded are rendered independent. That is, the joint probability of the recorded details and the hidden incident class read from the graph is

$$p(incident\text{-}class)\, p(airspace|incident\text{-}class)\, p(controller|incident\text{-}class)$$

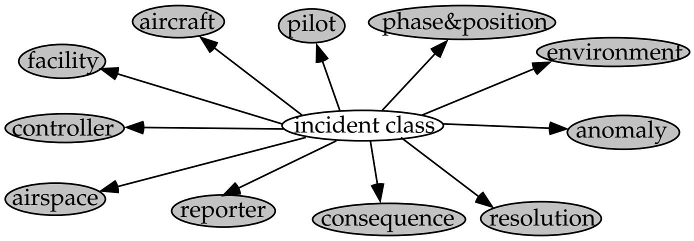

Figure 4.3
Simple unsupervised model of the aircraft incident domain.

$$p(facility|incident\text{-}class)\, p(aircraft|incident\text{-}class) \ldots$$

Each of these probabilities is evaluated using parameters set by the learning algorithm. For instance, a particular hidden incident class might have predominantly wide-body aircraft, experienced pilots, and have equipment failure, but otherwise the details are similar to the general population of incidents. The occurrence of wide-body aircraft, experienced pilots, and equipment failure would occur independently in this class, as indicated by the figure.

Aviation psychologists experienced in this domain expected relationships, for instance, between the pilot's qualifications and the type of aircraft, the type of aircraft and the phase of flight: for instance, wide-body aircraft do not go on joy rides. In some cases, these relationships where encoded as requirements of the Federal Aviation Authority, and in other cases they were well understood causal relationships. The discovered classes of aircraft incidents tended to be confounded by these known relationships. A way around this problem is to construct a hybrid model as given in Figure 4.4. The expected relationships are encoded into the model. For instance, that the pilot's qualifications are influenced by the aircraft, and that the facility tracking the aircraft depends on the type of aircraft and which airspace it is in (commercial, private and military aircraft have different behaviors) are encoded. This leaves the hidden *incident class* to explain the remaining regularity in the domain. That is, probability tables would be elicited from the aviation psychologists for the understood probability relations such as $p(controller|facility, aircraft)$ and these fixed in the model. The learning system now needs to refine the model by filling in the remaining parts of the model that are left unspecified by this knowledge elicitation.

Again, there are a number of interesting questions about this refinement approach. How can the refinement algorithm proceed with some parts of model fixed? This is not a difficult problem in the sense that standard algorithm schemes like the expectation max-

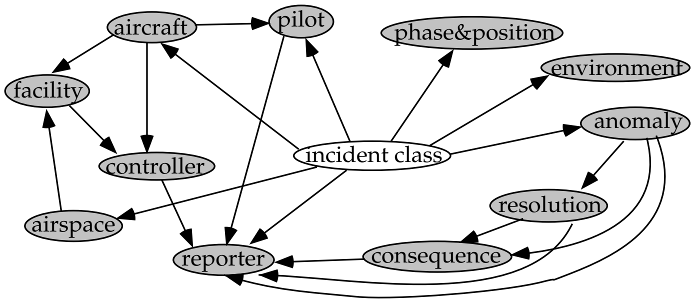

Figure 4.4
Hybrid unsupervised model of the aircraft incident domain.

imization (EM) algorithm used in SNOB and Autoclass are known to handle learning in this context (Buntine 1994). Software suited to this exact task is not currently available, however. So on this problem the iterative refinement process of knowledge discovery stops after one iteration, due to lack of available software.

4.5 Models for Learning and Discovery

This section outlines how various learning and discovery representations can be modeled with probabilistic graphical models. A characteristic problem is given along with the graphical model. The intention is to illustrate the rich variety of discovery tasks that can be represented with graphical models. Given the generality of the language, it should be clear that many hybrid models are represented as well, such as the hybrid unsupervised model of Figure 4.4.

The graphical models given here have their model parameters as well as the problem inputs marked as known. Of course, in the practice of data analysis, the model parameters are unknown and need to be learned from the data, and the training set or sample will usually have both problem inputs and outputs known for each case in the set. However, this represents the subsequent inference task underlying the problem, not the learning problem itself. In some cases, the functional form is also given for the probabilistic model implied by the graphical model.

4.5.1 Linear regression

Linear regression is the classic method in statistics for doing curve fitting, that is, predicting a real valued variable from input variables, real or discrete. See Casella and Berger

(1990), for instance, for a standard undergraduate introduction. Linear regression, in its most general form, fits non-linear curves as well because the term "linear" implies that the mean prediction for the variable is a linear function of the parameters of the model, but can be a non-linear function of the input variables. In the standard model, a Gaussian error function with constant standard deviation is used. This is shown in Figure 4.5. This is an instance of a generalized linear model (McCullagh and Nelder

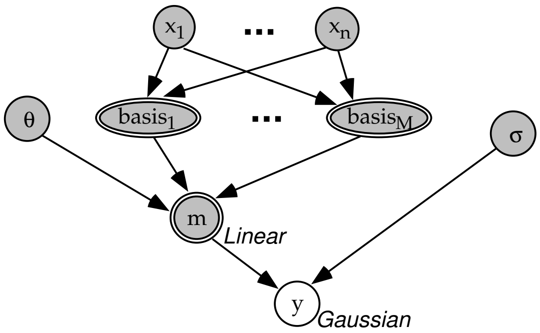

Figure 4.5
Linear regression with Gaussian error.

1989), so has a linear node at its core. The M basis functions $basis_1, \ldots, basis_M$ are known deterministic functions of the input variables $x_1, \ldots, x_n$. Variables that are deterministic functions of their inputs are represented with *deterministic nodes* that have double ellipses. These deterministic functions would typically be non-linear orthogonal functions such as Legendre polynomials. The linear node combines these linearly with the parameters θ to produce the mean m for the Gaussian.

$$m = \sum_{i=1}^{M} \theta_i basis_i(x) .$$

The graphical model of Figure 4.5 implies the above equation (each deterministic node implies an equality holds) and the conditional probability

$$p(y|x_1, \ldots, x_n, \theta, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-(y-m)^2/2\sigma^2} ,$$

the standard normal density with mean m and standard deviation σ. This graph also shows that the inputs x_1 to x_n are given, and so there is no particular distribution for them.

4.5.2 Weighted rule-based systems

Weighted rule-based systems are an interesting representation because they have been independently suggested in artificial intelligence, neural networks, and statistics, with each community using their own notation. The system, given in Figure 4.6, is the discrete version of the linear regression network given in Figure 4.5. Like linear regression, this is

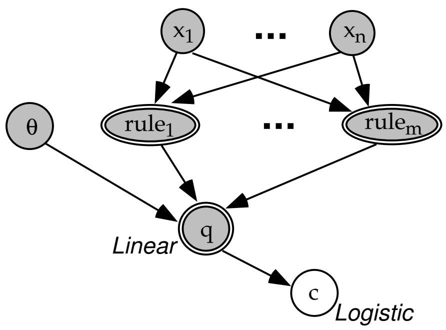

Figure 4.6
A weighted rule network.

also an instance of a generalized linear model, so has the linear construction of Figure 4.5 at its core. Each of the deterministic nodes for variables $rule_1, \ldots, rule_m$ represents a rule, an indicator function with the value 1 if the rule fires, and the value 0 otherwise. Those rules that fire cause weights (θ) to be added up, and consequently a prediction to be made.

In the binary classification case ($c \in \{0,1\}$), when multiple rules fire, probability that the class $c = 1$ is given by the transformation

$$p(c = 1|x_1, \ldots, x_n, \theta) \;=\; Logistic^{-1}\left(\sum_{i=1}^{m} rule_i \theta_i\right)$$

The functional type for the Logistic node is the function,

$$p(c = 1|u) \;=\; \frac{e^u}{1 + e^u} \;=\; 1 - Sigmoid(u) \;=\; Logistic^{-1}(u) \;, \qquad (4.5.4)$$

which maps a real value u onto a probability for the binary variable c. This function is the inverse of the logistic or logit function used in generalized linear models, and is also related to the sigmoid function used in feed-forward neural networks.

According to this weighting scheme, if a rule $rule_i$ fires in isolation, the probability that class $c = 1$ becomes $Logistic^{-1}(\theta_i)$. Hence θ_i can be interpreted as the log odds of p_i ($Logistic(p_i)$), where p_i is the probability that c will be 1 when only the single rule i fires. If multiple rules fire then this formula corresponds to combining the probabilities p_i using the original Prospector combining formula (Duda, Hart, and Nilsson 1976; Berka and Ivánek 1994)

$$Combine(p_i, p_j) \;=\; \frac{p_i \cdot p_j}{p_i \cdot p_j + (1 - p_i) \cdot (1 - p_j)} \;.$$

This combining formula is associative and commutative so the order of combination is irrelevant.

This approach thus implements a weighted rule-based system for classification using the Prospector combining formula. The model can also be interpreted as a neural network since the output node corresponds to a sigmoid, and the intermediate deterministic nodes can be interpreted as unparameterized hidden nodes. By using other combination rules different effects can be achieved; even for instance, fuzzy-style combinations.

4.5.3 Hierarchical mixtures of experts

Jordan and Jacobs (1993) have developed a classification approach based on the notion of a "mixture of experts". Like the weighted rule-based system, this model predicts a class c from a vector of inputs x. It does so, however, by combining a number of linear models to form a more complex classifier.

The decision tree representation and the DAG for this mixture model is given in the left and right of Figure 4.7 respectively. The decision tree is presented here for the case of discrete variables. In general both inputs and outputs can be real valued or discrete. Traversing the tree in the left of the figure down to a leaf node leads one to the leaf, which represents an "expert". These experts then combine to make the prediction for the class c. The prediction is done with a log-linear model, using the parameters $\mu' = \mu_{g_1 g_2}$, for the two "gates" g_1, g_2. Suppose the class is C-valued, so $c \in \{1, 2, \ldots, C\}$. The class prediction is:

$$p(c = i|x, \mu') \;=\; log\text{-}linear(i, x, \mu') \;=\; \frac{e^{x \cdot \mu'_i}}{\sum_{j=1}^{C} e^{x \cdot \mu'_i}} \;.$$

This is similar to the weighted rule-based system described in Section 4.5.2, where the rules correspond to the vector x. μ' is a matrix of dimension $C \times dim(x)$, and by convention $\mu'_C = 0$. For c binary, this is equivalent to the logistic node used in Section 4.5.2. The decision tree also has two variables denoting "gates", g_1 at the first node and g_2 at the two second level nodes, however, these are not present in the data. The values

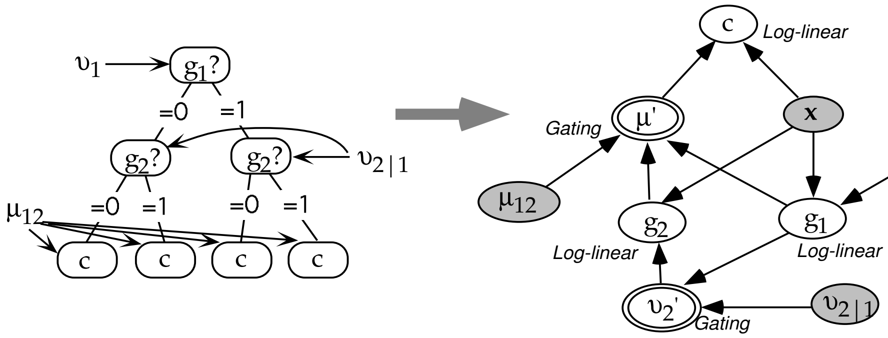

Figure 4.7
A two level mixture of "experts".

for the gates g_1 and g_2 are predicted using the data and the parameters υ_1 and $\upsilon_{2|g_1}$ respectively. At the first level is discrete valued gate g_1 (in the tree this is represented as binary, however, it can be N-ary in general). The first value is chosen in a probabilistic fashion according to the log-linear model with parameters υ_1.

$$p(g_1 = i|x, \upsilon_1) \;=\; \textit{log-linear}(i, x, \upsilon_1) \ .$$

υ_1 is a matrix of dimension $C \times dim(x)$, and by convention $\upsilon_{1,C} = 0$. A second gate g_2 is then chosen, again in a probabilistic fashion according to a log-linear model, but this time based on the first gate as well as the input x. The final probabilities for c are generated by another log-linear model as given by the first formula above.

In the graphical model this goes as follows. There are three log-linear models, two for the gates g_1 and g_2 and one for the final class probability. Gating nodes (which do matrix lookup) select the parameters for the log-linear nodes based on the values of other variables.

The graphical model of Figure 4.7 therefore yields the following conditional probability:

$$p(c|x, \upsilon_1, \upsilon_{2|1}, \mu_{12}) \;=\; \sum_{g_1} \textit{log-lin}(g_1, x, \upsilon_1) \sum_{g_2} \textit{log-lin}(g_2, x, \upsilon_{2|g_1})\, \textit{log-lin}(c, x, \mu_{g_1 g_2}) \ .$$

This is a *mixture* model (Titterington, Smith, and Makov 1985), in the sense that it sums over hidden variables g_1 and g_2, where the basic joint probability $p(c, g_1, g_2|x, \upsilon_1, \upsilon_{2|1}, \mu_{12})$ is in a standard form. If only one layer were used (so g_2 and associated gates were deleted), then this model corresponds to a supervised version of the unsupervised Autoclass system described next in Section 4.5.4.

4.5.4 Unsupervised learning

There are a range of unsupervised learning systems in statistics, neural networks, and artificial intelligence. Many of these can be represented as graphical models with hidden nodes that are used to represent hidden classes. In a sense, the learning of Bayesian networks from data can be called unsupervised learning as well, however, it is more accurately termed model discovery. This is described by Heckerman (1995). The aviation safety model given in Figure 4.4 is a hybrid of these different kinds of models.

Consider Autoclass III and the probabilistic unsupervised learning systems it is based on. For instance, a simple Autoclass III classification for three boolean variables var_1, var_2 and var_3 has the parameterization ϕ, θ_1, θ_2 and θ_3 given in Figure 4.8. The

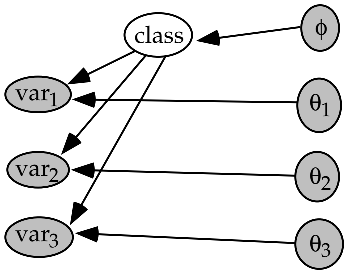

Figure 4.8
Explicit parameters for a simple Autoclass model.

class is unobserved or "hidden". If the class assignment where known, then the variables var_1, var_2 and var_3 would be rendered statistically independent, or "explained" in some sense. More complex models allow correlations between variables, but Autoclass III does not introduce this. The parameters ϕ (a vector of class probabilities) here gives the proportions for the hidden classes, and the three parameters θ_1, θ_2 and θ_3 give how the variables are distributed within each hidden class. For instance, if there are 10 classes, then ϕ is a vector of 10 class probabilities such that the prior probability for a case being in class c is ϕ_c. If var_1 is a binary variable, then θ_1 would be 10 probabilities, one for each class, such that if the case is known to be in class c, then the probability var_1 is true is given by $\theta_{1,c}$ and the probability var_1 is false is given by $1 - \theta_{1,c}$.

There are many other models for unsupervised learning that can be similarly represented with probabilistic graphs. Sometimes this includes undirected graphs or mixtures of directed and undirected graphs (Buntine 1994). This includes the stochastic networks used in Hopfield models and others in neural networks Hertz, Krogh, and Palmer (1991), more complex unsupervised learning systems such as Autoclass IV which has a variety of covariances (Hanson, Stutz, and Cheeseman 1991), and systems with multiple classes.

4.6 Learning algorithms

Methods have been developed for learning simple discrete and Gaussian Bayesian networks from data, and for learning simple unsupervised models such as those mentioned in Section 4.5.4. Given that all the previous models such as linear regression and weighted rule-based systems can also be represented as Bayesian networks, will these same learning algorithms apply? Unfortunately not. However, there are general categories of algorithm schemes for learning that can be mixed and matched to these various problems. Four categories considered here are represented by the models they address, given in Figure 4.9. This section briefly explains these categories. Algorithms for learning them are described

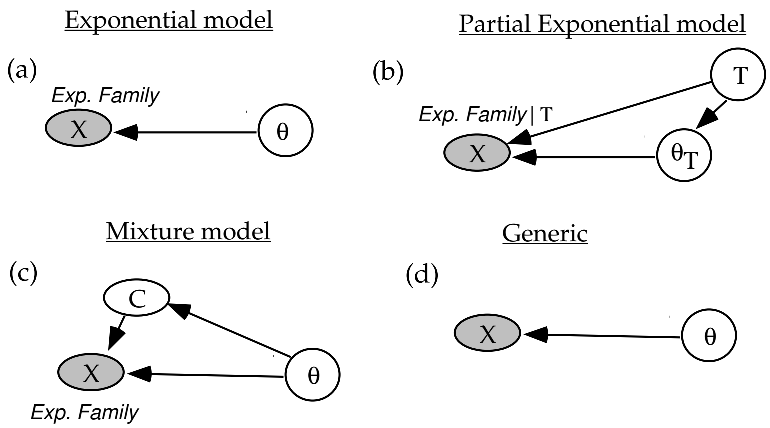

Figure 4.9
Four categories of models.

in (Buntine 1994), and references therein.

The simplest category of learning models have exact, closed form solutions to the learning problem. This category is the exponential family of distributions, which includes the Gaussian, the multinomial, and other basic distributions (Bernardo and Smith 1994), but also the decision tree or Gaussian Bayesian network of known fixed structure, and linear regression with Gaussian error described in Section 4.5.1. These exponential family distributions all have closed form solutions to the learning problem which are linear in the sample size (Bernardo and Smith 1994; Buntine 1994). For instance, if X has a univariate Gaussian distribution, then we estimate its unknown mean and standard deviation from the sample mean and sample standard deviation (usually along with some adjustment to make the estimate unbiased). No search or numerical optimization is involved. The exponential family category is represented by the *exponential model* in Figure 4.9(a). The probability model for the data given the parameters, $p(X|\theta)$ is shown in this figure to be in the exponential family.

Two important categories of learning models are based on the exponential family category. The second category of learning models is where a useful subset of the model does fall into the exponential family. This is represented by the *partial exponential model* in Figure 4.9(b). The part of the problem that is exponential family can be solved in closed form, as mentioned above. The remaining part of the problem is typically handled approximately. Decision trees and Bayesian networks over multinomial or Gaussian variables fall into second category (Buntine 1991a; Buntine 1991b; Spiegelhalter *et al.* 1993) when the structure of the tree or network is *not* known, as does linear regression with subset selection of relevant variables. In the figure, this is represented as follows. If we know the structure T, then the model is in the exponential family with parameters θ_T. So the probability model $p(X|\theta_T, T)$ is in the exponential family if we hold T fixed.

The third category of learning models is where, if some hidden variables are introduced into the data, the problem becomes exponential family if the hidden values were known. This is represented by the *mixture model* in Figure 4.9(c). In general, this family of models has that $p(X|C,\theta)$ is in the exponential family where C is the hidden variable (or variables) and θ are the model parameters. C does not occur in the data so this yields a probability model for X given by:

$$p(X|\theta) \;=\; \sum_C p(X|C,\theta)\, p(C|\theta) \;.$$

Two examples of this category are the mixture of experts model of Section 4.5.3, and the unsupervised learning models mentioned in Section 4.5.4. This category of models are used to model unsupervised learning, incomplete data in the classification problems, robust regression, and general density estimation (Titterington *et al.* 1985). The mixture model category can often be learned using the EM algorithm. The EM algorithm has an inner loop using the closed form solution found for the underlying exponential family model.

The final category of problems is a catch-all represented by the *generic model* in Figure 4.9(d). In this case the data X has the unconstrained probability model $p(X|\theta)$, and we assume nothing about its form. This includes feed-forward neural networks and the weighted rule-based model of Section 4.5.2. These models can be learned by algorithms such as the maximum *a posteriori* (MAP) algorithm and other general error minimization schemes. Notice that in general the other three categories of learning models can be cast into this form by ignoring some structural detail of the model. Hence the algorithms like the MAP algorithm can be applied to all the other categories of learning models as well.

4.7 Conclusion

The graphical component of the probabilistic models presented here is only relevant as a visual aid for describing models. However, the graphs provide a structural view of a probability model without getting lost in the mathematical detail. This is invaluable in the same way that a qualitative physical model can be invaluable for explaining behavior without recourse to the numeric detail. So what of probabilistic modeling? What does all this buy you?

First, probabilistic models provide a language for performing problem decomposition and recomposition, illustrated in Section 4.3, and knowledge refinement, illustrated in Section 4.4. Inference on the probabilistic models developed can be performed using a variety of probabilistic inference schemes, as listed in Section 4.2.

Second, because of the flexibility of probabilistic graphical models, they are a suitable language to represent a wide variety of learning models. Of course, the same can be said of C++. However, probabilistic models allow probability theory to be applied directly to derive inference algorithms via principles such as maximum likelihood, maximum *a posterior*, and other probabilistic schemes. Some relevant algorithms are discussed in Section 4.6. This offers a unifying conceptual framework for the developer, with, for instance, smooth transitions into other modes of probabilistic reasoning such as diagnosis, explanation, and information gathering.

Third, this probabilistic framework offers a computational approach to developing learning and discovery algorithms. The conceptual framework for this is given in Figure 4.10. Probability and decision theory are used to decompose a problem into a computational prescription, and then search and optimization techniques are used to fill the prescription. A software tool exists that implements a special case of this conceptual framework using Gibbs sampling as the computational scheme (Gilks *et al.* 1993). The Gibbs sampler is but one family of algorithms, and many more can be fit into this general framework. As explained by Buntine (1994), the framework of Figure 4.10 can use the categories of learning models described in Section 4.6 as its basis.

The real gain from the scheme of Figure 4.10 does not arise from the potential re-implementation of existing software, but from understanding gained by putting different models for learning and discovery in a common language, the ability to create novel hybrid

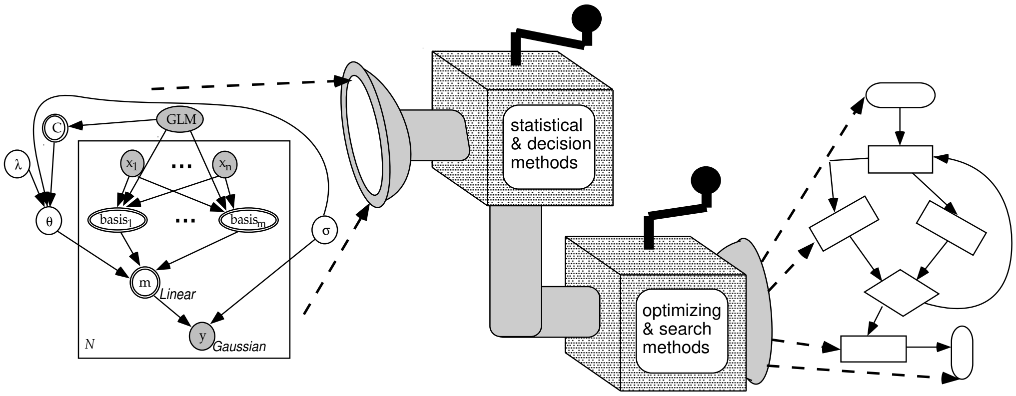

Figure 4.10
A software generator.

algorithms, and the ability to tailor special purpose algorithms for specific problems. For instance, by recognizing the connection between logistic regression, neural networks and Prospector rules, done in Section 4.5.2, we are able to borrow algorithms from other fields to address the task. The scheme of Figure 4.10 supports the problem decomposition and iterative knowledge refinement processes described in Sections 4.3 and 4.4.

Bibliography

Berka, P., and Ivánek, J. 1994. Automated knowledge acquisition for PROSPECTOR-like expert systems. In *Proceedings of the European Conference on Machine Learning*, 339–342.

Bernardo, J.M., and Smith, A.F.M. 1994. *Bayesian Theory*. Chichester: John Wiley.

Boulton, D.M., and Wallace, C.S. 1990. A program for numerical classification. *The Computer Journal*, 13(1):63–69.

Brachman, R.J., and Anand, T. 1995. The process of knowledge discovery in databases: A first sketch. In *Advances in Knowledge Discovery and Data Mining*, eds. U.M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R.S. Uthurasamy. MIT Press.

Buntine, W.L. 1994. Operations for learning with graphical models. *Journal of Artificial Intelligence Research*, 2:159–225.

Buntine, W.L. 1991a. Learning classification trees. In *Artificial Intelligence Frontiers in Statistics*, ed. D.J. Hand, 182–201. London: Chapman & Hall.

Buntine, W.L. 1991b. Theory refinement of Bayesian networks. In *Uncertainty in Artificial Intelligence: Proceedings of the Seventh Conference*, eds. B.D. D'Ambrosio, P. Smets, and P.P. Bonissone, 52–60. San Mateo, California: Morgan Kaufmann.

Buntine, W.L., 1990. Myths and legends in learning classification rules. In *Eighth National Conference on Artificial Intelligence*, 736–742. Boston, Massachusetts: AAAI Press.

Casella G., and Berger, R.L. 1990. *Statistical Inference*. Belmont, California: Wadsworth & Brooks/Cole.

Chan, B.Y. and Shachter, R.D. 1992. Structural controllability and observability in influence diagrams. In *Uncertainty in Artificial Intelligence: Proceedings of the Eight Conference*, eds. D. Dubois, M.P. Wellman, B.D. D'Ambrosio and P. Smets, 25–32. Stanford, California: Morgan Kaufmann.

Cheeseman, P., and Stutz, J. 1995. Bayesian clustering. In *Advances in Knowledge Discovery and Data Mining*, eds. U.M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R.S. Uthurasamy. MIT Press.

Dagum, P., Galper, A., Horvitz, E., and Seiver, A. 1995. Uncertain reasoning and forecasting. *International Journal of Forecasting*. Forthcoming.

Dean T.L., and Wellman, M.P. 1991. *Planning and Control*. San Mateo, California: Morgan Kaufmann.

Duda, R.O., Hart, P.E., and Nilsson, N.J. 1976. Subjective Bayesian methods for rule-based inference systems. In *National Computer Conference (AFIPS Conference Proceedings, Vol. 45)*, 1075–1082.

Fayyad, U.M., Djorgovski, S., and Weir, N. 1995. The SKICAT system for sky survey cataloging and analysis. In *Advances in Knowledge Discovery and Data Mining*, eds. U.M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R.S. Uthurasamy. MIT Press.

Gilks, W.R., Thomas, A., and Spiegelhalter, D.J. 1993. A language and program for complex Bayesian modelling. *The Statistician*, 43:169–178.

Ginsberg, A., Weiss, S.M., and Politakis, P. 1988. Automatic knowledge base refinement for classification systems. *Artificial Intelligence*, 35(2):197–226.

Hanson, R., Stutz, J., and Cheeseman, P. 1991 Bayesian classification with correlation and inheritance. In *International Joint Conference on Artificial Intelligence*, 692–698. San Mateo, California: Morgan Kaufmann.

Heckerman, D. 1995. Bayesian networks for knowledge representation and learning. In *Advances in Knowledge Discovery and Data Mining*, eds. U.M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R.S. Uthurasamy. MIT Press.

Heckerman, D. 1990. Probabilistic similarity networks. *Networks*, 20:607–636.

Henrion, M., Breese, J.S., and Horvitz, E.J. 1991. Decision analysis and expert systems. *AI Magazine*, 12(4):64–91.

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

Jordan, M.I., and Jacobs, R.I. 1993. Supervised learning and divide-and-conquer: A statistical approach. In *Machine Learning: Proc. of the Tenth International Conference*, 159–166. San Mateo, California: Morgan Kaufmann.

Kjæruff, U. 1992. A computational scheme for reasoning in dynamic probabilistic networks. In *Uncertainty in Artificial Intelligence: Proceedings of the Eight Conference*, eds. D. Dubois, M.P. Wellman, B.D. D'Ambrosio and P. Smets, 121–129. San Mateo, California: Morgan Kaufmann.

Kraft, P., and Buntine, W.L. 1993. Initial exploration of the ASRS database. In *Seventh International Symposium on Aviation Psychology*, Columbus, Ohio.

Lauritzen, S.L., Dawid, A.P., Larsen, B.N., and Leimer, H.-G. 1990. Independence properties of directed Markov fields. *Networks*, 20:491–505.

McCullagh, P., and Nelder, J.A. 1989. *Generalized Linear Models*. London: Chapman and Hall. Second edition.

Madigan, D., Mosurski, K., and Almond, R.G. 1995. Explanation in belief networks. StatSci research report 33, StatSci/Mathsoft, Seattle, Washington. (Submitted for publication.)

Matheus, C., Piatetsky-Shapiro, G., and McNeill, D. 1995. Key findings reporter for the analysis of healthcare information. In *Advances in Knowledge Discovery and Data Mining*, eds. U.M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R.S. Uthurasamy. MIT Press.

Ousterhout, J.K. 1994. *Tcl and the Tk Toolkit*. Addison-Wesley.

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

Poland, W.B. 1994. *Decision Analysis with Continuous and Discrete Variables: A Mixture Distribution Approach*. Ph.D. diss., Dept. of Engineering Economic Systems, Stanford Univ.

Rabiner, L.R., and Juang, B.H. 1986. An introduction to hidden Markov models. *IEEE ASSP Magazine*, January:4–16, 1986.

Shachter, R.D., Andersen, S.K., and Szolovits, P. 1994. Global conditioning for probabilistic inference in belief networks. In *Uncertainty in Artificial Intelligence: Proceedings of the Tenth Conference*, eds, R. Lopez de Mantaras and D. Poole, 514–522. San Mateo, California: Morgan Kaufmann.

Shachter, R.D., and Heckerman, D. 1987. Thinking backwards for knowledge acquisition. *AI Magazine*, 8(Fall):55–61.

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

Smyth, P, and Mellstrom, J. 1992. Detecting novel classes with applications to fault diagnosis. In *Ninth International Conference on Machine Learning*. San Mateo, California: Morgan Kaufmann.

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

Titterington, D.M., Smith, A.F.M., and Makov, U.E. 1985. *Statistical Analysis of Finite Mixture Distributions*. Chichester: John Wiley & Sons.

Towell, G.G., Shavlik, J.W., and Noordewier, M.O. 1990. Refinement of approximate domain theories by knowledge-based neural networks. In Eighth National Conference on Artificial Intelligence, 861–866. Boston, Massachusetts: AAAI Press.

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