CONSTRUCTION OF A MARKOV NETWORK FROM DATA FOR PROBABILISTIC INFERENCE

S.K.M Wong and Y. Xiang
Department of Computer Science
University of Regina
Regina, Saskatchewan, Canada S4S 0A2
E-mail: wong@cs.uregina.ca, yxiang@cs.uregina.ca

Keywords: Knowledge discovery, Learning, probabilistic inference, Markov networks.

ABSTRACT

We consider automatic construction of a Markov network as an alternative to learning classification rules. A Markov network consists of an undirected graph as a qualitative domain model and a factorized probability distribution on the graph. We present a procedure for constructing a Markov network from a sample set of observations. The procedure minimizes the Kullback-Leibler cross-entropy between the network under construction and the set of samples in a stepwise fashion, until a preset threshold is reached.

We present our experimental results as well as how the learned network can be used in probabilistic inference.

INTRODUCTION

In automatic construction of many knowledge-based systems, the input is represented as a table of columns (variables, attributes) and rows (tuples). Each row may be viewed as a single instance of observation. Thus, the input consists of a set of observed instances. For example, each tuple may represent a group of symptoms and diseases that a patient has. From the vantage point of inductive learning, the main task is to develop a method for constructing *inference rules* from a sample set of observations. These rules can then be applied to classify future cases (unobserved instances).

Many techniques have been developed over the years for constructing *explicit* decision rules from a set of samples such as generation of classification trees (Quinlan 1986) and computation of reducts (Pawlak 1991). Alternatively, one may construct a belief network from a sample of observations for probabilistic inference (Herskovits and Cooper 1990; Cooper and Herskovits 1992; Pittarelli 1990). In contrast to the methods for generating explicit classification rules, belief networks attempt to capture the relationships among a set of variables without the need to designate a decision (expert) variable. Once a belief network is constructed, we can compute the probability of any subset of variables conditioned on any other subset. This means that, with a belief network, we can make predictions based on the available values of any subset of variables. On the other hand, with a rule-based system, the value of every variable in a rule must be known before the rule can be applied.

This paper suggests an entropy-based procedure for constructing a Markov network. The algorithm is based on the Kullback-Leibler cross-entropy (kullback and Leibler 1951) which allows us to choose a full joint probability distribution in the absence of complete information about the exact distribution.

A MARKOV NETWORK

Before discussing Markov networks, let us first introduce the notion of *hypertrees* (Shafer 1991).

Let $\mathcal{L}$ denote a lattice. We say that $\mathcal{H}$ is a *hypergraph*, if $\mathcal{H}$ is a finite subset of $\mathcal{L}$. Consider, for example, the power set $2^{\mathcal{X}}$, where $\mathcal{X} = \{x_1, x_2, ..., x_n\}$ is a set of variables. The power set $2^{\mathcal{X}}$ is a lattice of all subsets of $\mathcal{X}$. Any subset of $2^{\mathcal{X}}$ is a hypergraph on $2^{\mathcal{X}}$. We say that an element t in a hypergraph $\mathcal{H}$ is a *twig* if there exists another element b in $\mathcal{H}$, distinct from t, such that $t \cap (\cup(\mathcal{H} - \{t\}) = t \cap b$. We call any such b a *branch* for the twig t. A hypergraph $\mathcal{H}$ is a *hypertree* if its elements can be ordered, say $h_1, h_2, ..., h_i$, so that h_i is a twig in $\{h_1, h_2, ..., h_i\}$, for $i = 2, ..., n$. We call any such ordering a *hypertree construction ordering* for $\mathcal{H}$. Given a hypertree construction ordering $h_1, h_2, ..., h_n$, we can choose, for i from 2 to n, an integer $b(i)$ such that $1 \leq b(i) \leq i-1$ and $h_{b(i)}$ is a branch for h_i in $\{h_1, h_2, ..., h_i\}$. We call the function $b(i)$ satisfying this condition a branch function for $\mathcal{H}$ and $h_1, h_2, ..., h_n$.

For example, let $\mathcal{X} = \{x_1, x_2, ..., x_6\}$ and $\mathcal{L} = 2^{\mathcal{X}}$. Consider a hypergraph, $\mathcal{H} = \{h_1 = \{x_1, x_2, x_3\}, h_2 = \{x_1, x_2, x_4\}, h_3 = \{x_2, x_3, x_5\}, h_4 = \{x_5, x_6\}\}$, depicted in Figure 1. This hypergraph is in fact a hypertree; the sequence, h_1, h_2, h_3, h_4, is a hypertree construction ordering. Furthermore, $b(2) = 1, b(3) = 1$, and $b(4) = 3$.

Consider a joint probability distribution defined as follows:

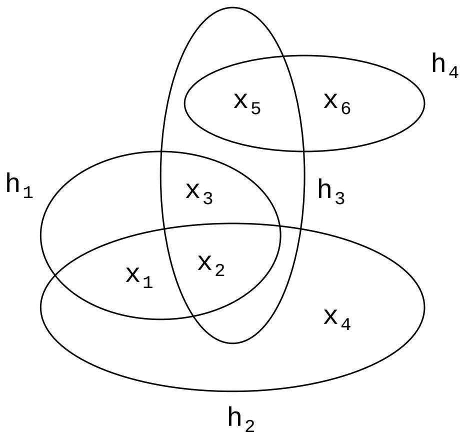

Figure 1: A graphical representation of the hypergraph $\mathcal{H} = \{h_1 = \{x_1, x_2, x_3\}, h_2 = \{x_1, x_2, x_4\}, h_3 = \{x_2, x_3, x_5\}, h_4\{x_5, x_6\}\}$.

$$p(\mathbf{c}) = p(c_1, c_2, c_3, c_4, c_5, c_6) = \tfrac{p(c_1,c_2,c_3)p(c_1,c_2,c_4)p(c_2,c_3,c_5)p(c_5,c_6)}{p(c_1,c_2)p(c_2,c_3)p(c_5)} \quad (1)$$

where $\mathbf{c} = (c_1, c_2, c_3, c_4, c_5, c_6)$ is a *configuration* of the set of variables $\mathcal{X} = \{x_1, x_2, x_3, x_4, x_5, x_6\}$. The dependencies among variables in the distribution can be depicted as an undirected graph as shown in Figure 2.

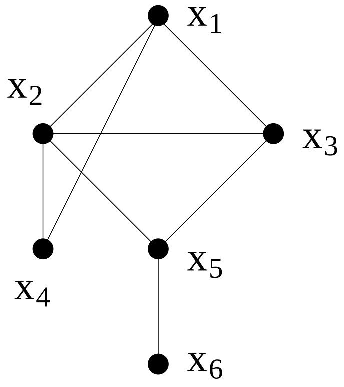

Figure 2: An undirected graph representing a Markov network.

In many applications, it is more convenient to characterize the dependencies by a hypergraph. The hypergraph corresponding to the distribution $p(\mathbf{c})$ defined by Equation 1 is shown in Figure 1 in which each hyperedge is a *maximal clique* of the undirected graph in Figure 2. We say that $p(\mathbf{c})$ is *factorized* on such a hypergraph. A joint probability distribution is called a Markov distribution (Pearl 1988; Hajek *et al.* 1992), if it is factorized on a hypertree.

CONSTRUCTION OF A MARKOV NETWORK

In this section, we describe a procedure for constructing a Markov network from a set of observed instances.

Ideally, based on the observed data, one would like to find a Markov distribution that produces maximum entropy as such a distribution has the least bias (Jaynes 1982). However, the computational complexity for searching this distribution is too high. So far, no efficient algorithm has yet been found for this task.

In this paper, similar to the method suggested by Chow and Liu (1968), we adopt the Kullback-Leibler cross-entropy as a measure of *closeness* between two probability distributions:

$$I(p, p') = \sum_{\mathbf{c}} p(\mathbf{c}) \log \frac{p(\mathbf{c})}{p'(\mathbf{c})}, \quad (2)$$

where $\mathbf{c} = (c_1, c_2, \ldots, c_n)$ is a configuration of the set of variables $\mathcal{X} = \{x_1, x_2, \ldots, x_n\}$. With a fixed p, we can choose, from the set of all possible Markov distributions $\{p'\}$, the distribution p_0 that minimizes the cross-entropy $I(p, p')$. For a Markov distribution p', Equation 2 can be expressed as:

$$\begin{aligned} I(p, p') &= \sum_{\mathbf{c}} p(\mathbf{c}) \log p(\mathbf{c}) - \sum_{\mathbf{c}} p'(\mathbf{c}) \log p'(\mathbf{c}) \\ &= H(p') - H(p). \end{aligned}$$

Thus, for a fixed p, minimizing the closeness metric (i.e., the cross-entropy) among the elements in $\{p'\}$ is equivalent to minimizing the entropy $H(p')$, namely:

$$\min_{p'' \in \{p'\}} (I(p, p'')) = \min_{p'' \in \{p'\}} (H(p'')). \quad (3)$$

An *approximate* method for constructing the desired full joint distribution is outlined as follows. Initially, we may assume that all variables are probabilistically independent, i.e., there exists no edge between any two nodes (variables) in the undirected graph representing the distribution. Then an edge is added to the graph subject to the restriction that the resultant hypergraph must be a hypertree. The undirected graph corresponding to the distribution with minimum entropy is being selected as the graph for further addition of other edges. This process is repeated until a threshold is reached in the rate of decrease of entropy between successive Markov distributions.

In practice, it may not be feasible to compute the entropy directly by using the full joint distribution. It is necessary to use a more efficient formula than Equation 2 for the computation, as the number of times that one has to compute the entropy, during the search, could be very large. It was shown in (Wong 1994) that the entropy of a Markov distribution p can be expressed as:

$$H(p) = \sum_{i=1}^{n} H(p(h_i)) - \sum_{j=2}^{n} H(p(h_j \cap h_{b(j)})), \quad (4)$$

where h_i is a hyperedge in the hypergraph $\mathcal{H}$ representing the joint distribution p, $H(p(h_i))$ is the entropy of the marginal distribution $p(h_i)$, the sequence $h_1, h_2, \ldots, h_n$ is a hypertree construction ordering for $\mathcal{H}$, and $b(i)$ for $2 \leq i \leq n$ is the branch fucntion for this particular ordering. It should be noted that all the required marginal distributions are estimated from the input set of observed data. It is not difficult to see that Equation 4 is a much more efficient formula for computing the entropy of a Markov distribution.

EXPERIMENTAL RESULTS

To evaluate our method, we first discuss the learning of a Markov network from a full joint probability distribution (jpd) (equivalent to an infinite number of samples). Then learning from a finite set of sample observations is discussed.

p(tamp	&alar	&leav	&rept	&fire	&smok) =	.000059400
p(tamp	&alar	&leav	&rept	&fire	&¬smok) =	.000006600
p(tamp	&alar	&leav	&rept	&¬fire	&smok) =	.000111078
p(tamp	&alar	&leav	&rept	&¬fire	&¬smok) =	.010996723
p(tamp	&alar	&leav	&¬rept	&fire	&smok) =	.000019800
p(tamp	&alar	&leav	&¬rept	&fire	&¬smok) =	.000002200
p(tamp	&alar	&leav	&¬rept	&¬fire	&smok) =	.000037026
p(tamp	&alar	&leav	&¬rept	&¬fire	&¬smok) =	.003665574
p(tamp	&alar	&¬leav	&rept	&fire	&smok) =	.000000108
p(tamp	&alar	&¬leav	&rept	&fire	&¬smok) =	.000000012
p(tamp	&alar	&¬leav	&rept	&¬fire	&smok) =	.000000202
p(tamp	&alar	&¬leav	&rept	&¬fire	&¬smok) =	.000019994
p(tamp	&alar	&¬leav	&¬rept	&fire	&smok) =	.000010692
p(tamp	&alar	&¬leav	&¬rept	&fire	&¬smok) =	.000001188
p(tamp	&alar	&¬leav	&¬rept	&¬fire	&smok) =	.000019994
p(tamp	&alar	&¬leav	&¬rept	&¬fire	&¬smok) =	.001979410
p(tamp	&¬alar	&leav	&rept	&fire	&smok) =	.000001350
p(tamp	&¬alar	&leav	&rept	&fire	&¬smok) =	.000000150
p(tamp	&¬alar	&leav	&rept	&¬fire	&smok) =	.000000446
p(tamp	&¬alar	&leav	&rept	&¬fire	&¬smok) =	.000044105
p(tamp	&¬alar	&leav	&¬rept	&fire	&smok) =	.000000450
p(tamp	&¬alar	&leav	&¬rept	&fire	&¬smok) =	.000000050
p(tamp	&¬alar	&leav	&¬rept	&¬fire	&smok) =	.000000148
p(tamp	&¬alar	&leav	&¬rept	&¬fire	&¬smok) =	.000014701
p(tamp	&¬alar	&¬leav	&rept	&fire	&smok) =	.000000882
p(tamp	&¬alar	&¬leav	&rept	&fire	&¬smok) =	.000000098
p(tamp	&¬alar	&¬leav	&rept	&¬fire	&smok) =	.000000291
p(tamp	&¬alar	&¬leav	&rept	&¬fire	&¬smok) =	.000028815
p(tamp	&¬alar	&¬leav	&¬rept	&fire	&smok) =	.000087318
p(tamp	&¬alar	&¬leav	&¬rept	&fire	&¬smok) =	.000009702
p(tamp	&¬alar	&¬leav	&¬rept	&¬fire	&smok) =	.000028815
p(tamp	&¬alar	&¬leav	&¬rept	&¬fire	&¬smok) =	.002852679
p(¬tamp	&alar	&leav	&rept	&fire	&smok) =	.005762987
p(¬tamp	&alar	&leav	&rept	&fire	&¬smok) =	.000640332
p(¬tamp	&alar	&leav	&rept	&¬fire	&smok) =	.000006403
p(¬tamp	&alar	&leav	&rept	&¬fire	&¬smok) =	.000633929
p(¬tamp	&alar	&leav	&¬rept	&fire	&smok) =	.001920996
p(¬tamp	&alar	&leav	&¬rept	&fire	&¬smok) =	.000213444
p(¬tamp	&alar	&leav	&¬rept	&¬fire	&smok) =	.000002134
p(¬tamp	&alar	&leav	&¬rept	&¬fire	&¬smok) =	.000211310
p(¬tamp	&alar	&¬leav	&rept	&fire	&smok) =	.000010478
p(¬tamp	&alar	&¬leav	&rept	&fire	&¬smok) =	.000001164
p(¬tamp	&alar	&¬leav	&rept	&¬fire	&smok) =	.000000012
p(¬tamp	&alar	&¬leav	&rept	&¬fire	&¬smok) =	.000001153
p(¬tamp	&alar	&¬leav	&¬rept	&fire	&smok) =	.001037338
p(¬tamp	&alar	&¬leav	&¬rept	&fire	&¬smok) =	.000115260
p(¬tamp	&alar	&¬leav	&¬rept	&¬fire	&smok) =	.000001153
p(¬tamp	&alar	&¬leav	&¬rept	&¬fire	&¬smok) =	.000114107
p(¬tamp	&¬alar	&leav	&rept	&fire	&smok) =	.000001323
p(¬tamp	&¬alar	&leav	&rept	&fire	&¬smok) =	.000000147
p(¬tamp	&¬alar	&leav	&rept	&¬fire	&smok) =	.000145384
p(¬tamp	&¬alar	&leav	&rept	&¬fire	&¬smok) =	.014393064
p(¬tamp	&¬alar	&leav	&¬rept	&fire	&smok) =	.000000441
p(¬tamp	&¬alar	&leav	&¬rept	&fire	&¬smok) =	.000000049
p(¬tamp	&¬alar	&leav	&¬rept	&¬fire	&smok) =	.000048461
p(¬tamp	&¬alar	&leav	&¬rept	&¬fire	&¬smok) =	.004797688
p(¬tamp	&¬alar	&¬leav	&rept	&fire	&smok) =	.000000864
p(¬tamp	&¬alar	&¬leav	&rept	&fire	&¬smok) =	.000000096
p(¬tamp	&¬alar	&¬leav	&rept	&¬fire	&smok) =	.000094985
p(¬tamp	&¬alar	&¬leav	&rept	&¬fire	&¬smok) =	.009403468
p(¬tamp	&¬alar	&¬leav	&¬rept	&fire	&smok) =	.000085572
p(¬tamp	&¬alar	&¬leav	&¬rept	&fire	&¬smok) =	.000009508
p(¬tamp	&¬alar	&¬leav	&¬rept	&¬fire	&smok) =	.009403469
p(¬tamp	&¬alar	&¬leav	&¬rept	&¬fire	&¬smok) =	.930943429

Table 1: The joint probability distribution of the fire-alarm problem.

Learning A Markov Network From A Full Joint Distribution

For demonstration purposes, we use, in our experiments, a small full jpd for the fire-alarm problem (Poole and Neufeld 1988) given in Table 1.

This simple distribution describes the events involved when fire or tampering occurs in a building. It is assumed that the alarm would be activated by the occurrence of either fire or tampering. If the alarm is on, people inside the building should leave the building and this evacuation is then reported to the security. It can be verified that the following conditional independencies hold in this jpd. We have capitalized each variable name to distinguish the variable from the possible values (lower case) of the variable.

$$p(Smoke|Fire\&Tampering\&Alarm\&Smoke\&Leaving\&Report)$$
$$= p(Smoke|Fire)$$
$$p(Alarm|Fire\&Tampering\&Smoke)$$
$$= p(Alarm|Fire\&Tampering)$$
$$p(Leaving|Fire\&Tampering\&Alarm\&Smoke)$$
$$= p(Leaving|Alarm)$$
$$p(Report|Fire\&Tampering\&Alarm\&Smoke\&Leaving)$$
$$= p(Report|Leaving)$$

The above conditional independencies can be conveniently represented by a Markov network as shown in Figure 3.

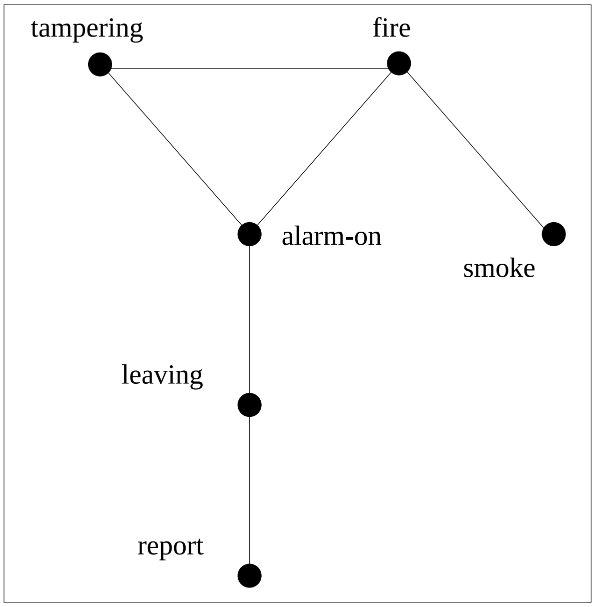

Figure 3: The Markov network of the fire-alarm joint distribution.

The main objective of our experiments is to check how well our method outlined in Section is able to learn the probabilistic conditional independencies. Learning starts with a completely disconnected Markov network. This means that all the variables are initially assumed to be probabilistically independent of each other. At each step of search, each possible single link is added to the current network, and the entropy of the resulting Markov network is computed. Note that for n unconnected nodes, there are $O(n^2)$ single links. The network yielding the lowest entropy is chosen to be the starting network entering into the next step of search. After each step of search, the decrement of entropy is checked against a predetermined threshold. If the decrement is less than the threshold, the learning process terminates.

The threshold should be set according to the size of the sample set, i.e., the smaller the sample set, the larger the threshold. The intuition is that, when the sample set is smaller, erroneous dependencies may be introduced and hence erroneous links are added to the network. Thus, by using a larger threshold, these *false* dependencies can be suppressed.

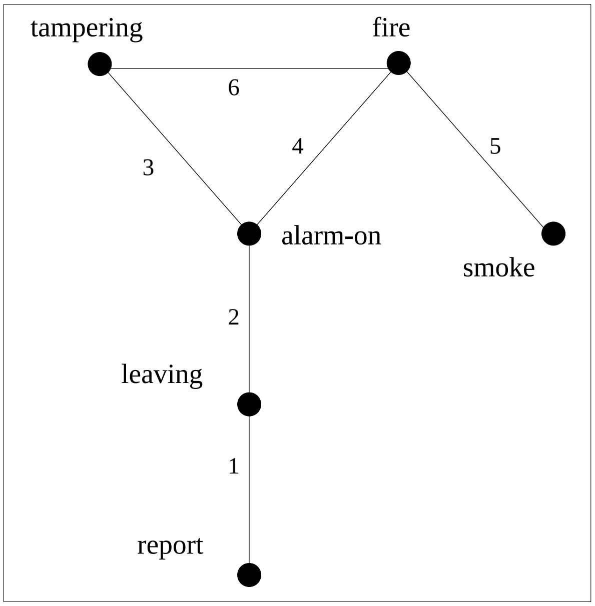

Figure 4: The learned Markov network in the fire-alarm domain.

When we use the full jpd for learning, we set the threshold at 0.001. Our method produced the *exact* Markov network as shown in Figure 4. The numbers in the figure indicate the order in which the links were added. The learning process terminated after the 6th link was added.

Learning A Markov Network From Samples

The second experiment is to test our method using a finite set of sample observations. We used the technique of logic sampling (Henrion 1988) to generate 1000 samples. The entropy threshold was set at 0.01. Again, the exact Markov network was produced by using such a sample set.

Using a finite set of samples, it is expected that the learned jpd would be different from the original one. Since the probability values of the jpd are generally small, we compared the exact clique marginals with those obtained from the learned jpd. The exact clique marginals range from 0.000098 to 0.98. The corresponding clique marginals in the learned Markov network range from 0 to 0.978. The maximal difference in probability value is 0.01495.

Using A Markov Network For Probabilistic Inference

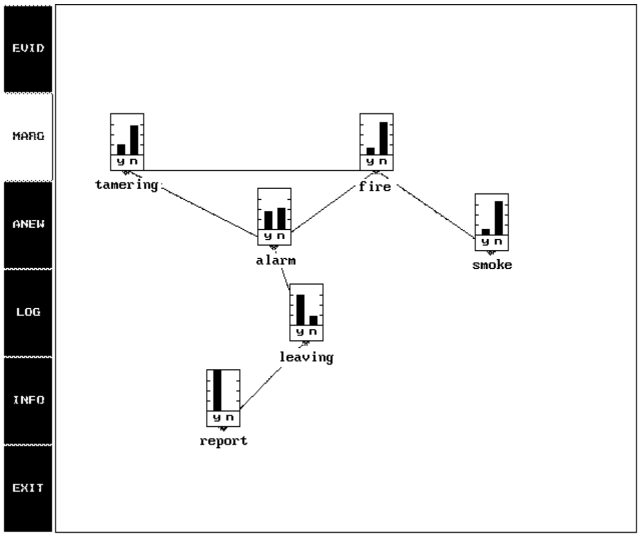

Figure 5: Probabilistic inference using the learned Markov network. Each histogram shows the probability distribution of the corresponding variable. The variable 'report' has been instantiated as 'yes' by evidence. Queries on all other unobserved variables can be answered.

One of the advantages of using a belief network in decision making over a set of classification rules (Ziarko 1991) or a decision tree (Quinlan 1986) is that a Markov network, for example, allows one to use any variable as a decision variable. For example, we can post to the fire-alarm network the query, "What is the probability of fire given that report is received?". We can post another query, "What is the probability of tampering given that report is received?", which involves a different decision variable. Figure 5 shows the answers to these queries, in the form of updated probability distributions, obtained from the same Markov network. In contrast, with conventional approaches, it would require the generation of different rules or a decision tree in order to answer both of these queries.

Another advantage of using a belief network for inference is that any number of observed variables can be used to obtain a probability on the decision variable. On the other hand, in rule-based systems, a rule can be fired only when all the conditional variables are observed. Figure 6 shows that the same Markov network can be used to answer the query, "What is the probability of fire given that both report is received and smoke is observed?".

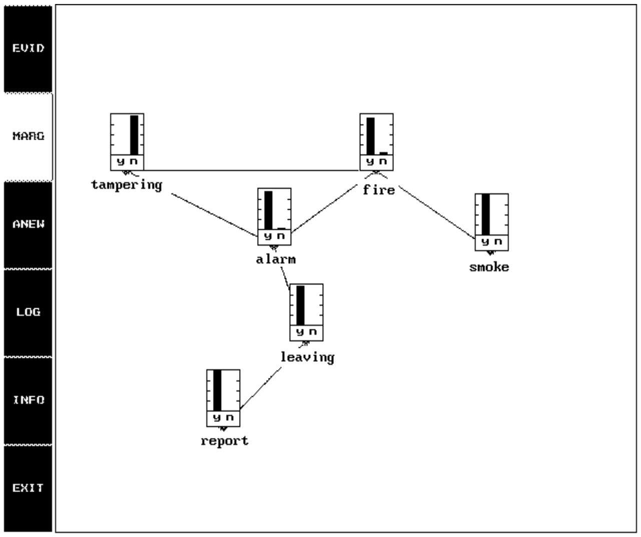

Figure 6: Probabilistic inference with varied evidence pattern. An additional variable 'smoke' is instantiated as 'yes'.

ACKNOWLEDGEMENT

This work is supported in part by Research Grant OGP0000982 and OGP0155425 from the Natural Sciences and Engineering Research Council (NSERC). The authors are also members of the Institute for Robotics and Intelligent Systems (IRIS) and wish to acknowledge the support of the Networks of Centres of Excellence Program of the Government of Canada, the NSERC and the participation of PRECARN Associates Inc.

References

[1] Chow, C.K. and C.N. Liu. 1968. "Approximating discrete probability distributions with dependence trees." *IEEE Trans. On Information Theory*, IT-14:462-467.

[2] Cooper, G.F. and E. Herskovits. 1992. "A Bayesian method for the induction of probabilistic networks from data." *Machine Learning*, (9):309–347.

[3] Hajek, P.; T. Hovranek and R. Jirousek. 1992. *Uncertain Information Processing in Expert Systems*, CRC Press.

[4] Henrion, M. 1988, "Propagation of uncertainty by probabilistic logic sampling in Bayes' networks." In *Proc. Sixth Conference on Uncertainty in Artificial Intelligence 2*, Amsterdam: Elsevier Science Publishers.

[5] Herskovits, E.H. and G.F. Cooper. 1990. "Kutato: an entropy-driven system for construction of probabilistic expert systems from database." In *Proc. Sixth Conf. Uncertainty in AI*, Cambridge, M.A., 54-62.

[6] Jaynes, E.T. 1982. "On the rationale of maximum-entropy methods." *Proc. of the IEEE*, 70 (9), 939-952.

[7] Kullback, S. and R.A. Leibler. 1951. "Information and sufficiency." *annals of Mathematical Statistics.* 22:79-86.

[8] Pawlak, Z. 1991. *Rough sets: theoretical aspects of reasoning about data*, Klwer Academic Publishers.

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

[10] Pittarelli, M. 1990. "Reconstructability analysis: An overview." *Revue Internationale de Systemique*, 4, 5-32.

[11] Poole, D. and E. Neufeld. 1988. "Sound probabilistic inference in Prolog: an executable specification of influence diagrams." In *I SIMPOSIUM INTERNACIONAL DE INTELIGENCIA ARTIFICIAL.*

[12] Quinlan, J.R. 1986. "Induction of decision trees." In *Machine Learning 1*, 81-106.

[13] Shafer, G. 1991. "An axiomatic study of computation in hypertrees." School of Business Working Paper Series (No. 232), University of Kansas.

[14] Wong, S.K.M. 1994. "The relational structure of belief networks." Submitted for publication.

[15] Ziarko, W. 1991. "The discovery, analysis, and representation of data dependencies in database." In *Knowledge Discovery in Databases*, G. Piatetsky-Sapiro and W.J. Frawley, eds. AAAI/MIT, 195-209.