Pp. 32–37 in *Proceedings of the Conference on Artificial Intelligence Research in Finland* (Turku, Finland, August 1994), edited by C.Carlsson, T.Järvi and T.Reponen. Finnish Artificial Intelligence Society, Helsinki 1994.

Learning Bayesian Prototype Trees by Simulated Annealing

Petri Myllymäki and Henry Tirri
University of Helsinki, Department of Computer Science
P.O.Box 26, FIN-00014 University of Helsinki, FINLAND
Petri.Myllymaki@cs.Helsinki.FI, Henry.Tirri@cs.Helsinki.FI

***Abstract*— Given a set of samples of an unknown probability distribution, we study the problem of constructing a good approximative Bayesian network model of the probability distribution in question. This task can be viewed as a search problem, where the goal is to find a maximal probability network model, given the data. In this work, we do not make an attempt to learn arbitrarily complex multi-connected Bayesian network structures, since such resulting models can be unsuitable for practical purposes due to the exponential amount of time required for the reasoning task. Instead, we restrict ourselves to a special class of simple tree-structured Bayesian networks called Bayesian prototype trees, for which a polynomial time algorithm for Bayesian reasoning exists. We show how the probability of a given Bayesian prototype tree model can be evaluated, given the data, and how this evaluation criterion can be used in a stochastic simulated annealing algorithm for searching the model space. The simulated annealing algorithm provably finds the maximal probability model, provided that a sufficient amount of time is used.**

I. Introduction

Learning structures from data is always relative to a class of models, set of structures which share some common properties. In our work we have restricted ourselves to learning probabilistic network structures, i.e., constructing a Bayesian network representation of a probability distribution described by a set of representative data samples. This learning task can be described in the abstract setting as a search in the network space, where the goal is to find a maximal probability network model for the data. The learning process has two important components: an evaluation criterion for a candidate network, and a search algorithm for selecting new candidate networks. Under certain assumptions, a probabilistic evaluation criterion for comparing different Bayesian network models can be constructed by computing directly the probability of a given Bayesian network [8], or indirectly [16] by using the Minimum Description Length approach [26]. This type of criteria are totally independent of the search algorithm used, and hence each search algorithm can be combined with several different criteria. In this paper, we use a combination of the techniques adopted from [8, 16, 26] for constructing an example of suitable evaluation criteria.

This research was supported by the Technology Development Center (TEKES).

In principle, the best model for the problem domain probability distribution can be found by going through all the possible networks, and choosing the model with the highest probability. Unfortunately, in the Bayesian network framework the number of possible models is too large for such exhaustive search approach to be used in practice (see the combinatorial analysis in [27]). For this reason most existing learning methods for Bayesian networks typically use simple greedy heuristics for searching the model space. In order to improve the performance of the learning algorithms, more elaborate search algorithms need to be used. Standard numerical optimization methods, such as gradient descent or conjugate gradient methods are not directly applicable since the gradient of the function to be maximized is not available. Another, more severe drawback with these methods is that there is nothing to prevent the methods from getting stuck in local maxima. This is an important aspect, as in the large model space in question the probability of having a large number of local maxima is very high, in which case the quality of the solution found with such optimization methods may be poor.

Simulated annealing [18] is an iterative stochastic sampling method which has been shown to be able to find a maximum probability state of the sampling space almost surely, provided that a sufficient number of iterations is used [9]. After being introduced to the optimization theory community in [12], simulated annealing has been applied to many different (NP-hard) optimization problems, such as TSP, graph partitioning, graph coloring, number set partitioning and clustering (for an extensive survey of applications, see [1, pp.89–90]). Unfortunately, the number of iterations for the theoretically guaranteed convergence is too high for practical purposes, but in many cases good results has been obtained even with relatively small number of iterations [1].

Simulated annealing has also been used for implementing the reasoning computation in the Bayesian networks [20]. However, as suggested in [5], in this work we use the simulated annealing algorithm for learning, i.e., finding a globally maximal probability Bayesian network model for our problem domain. In contrast to most existing learning algorithms for Bayesian networks, we do not try to learn arbitrarily complex multi-connected Bayesian network structures. In this restriction we are motivated by the fact that in the general case the resulting model can be unsuitable for practical purposes as the Bayesian reasoning task with multi-connected networks may take an exponential amount of time [7]. Instead, we restrict ourselves to a special class of simple tree-structured Bayesian networks, *Bayesian prototype trees* [22, 23], in which case a polynomial time algorithm for Bayesian reasoning exists [25]. In addition, the Bayesian prototype tree model can also be implemented extremely efficiently on massively parallel hardware, as there is a direct mapping from a Bayesian tree structure to a multi-layer feedforward neural network structure [22]. The model also conforms interestingly to the intuitively appealing *memory-based reasoning* paradigm (see e.g. [13]), and it can be seen as a Bayesian solution to the case matching and case adaptation problems (see [23]) in such domains.

The Bayesian prototype tree model is based on the assumption that the problem domain probability distribution can be approximated by a set of prototype vectors, where each prototype represents a set of (in some sense) similar problem domain instances. The approach is very similar to finite mixture models, and statistical kernel-based density estimators, where the problem domain probability distribution is approximated as a finite sum of particular types of kernel distributions (e.g., Gaussian). Such models are becoming increasingly popular in the *neural network* community, since the models can often be implemented very efficiently on massively parallel neural network architectures [19, 29, 17, 11]. In our approach we restrict ourselves to discrete problem domains, which allows us to drop all assumptions about the form of the underlying probability distribution. The prototype vectors are regarded as values of a latent hidden random clustering variable, which forms the root of the prototype tree model, and the actual problem domain random variables are the leaves of the tree. The approach resembles the discrete version of the *AutoClass* model [4], the *simple Bayesian classifier* model in [14] and [15], and the finite mixture model of multivariate Bernoulli distributions in [10], but unlike these models, our network can be used also for general regression tasks, not only for classification. It is important to notice that by introducing the artificial clustering variable, we do not have to limit the expressive power of our model, since it is obvious that theoretically the Bayesian prototype tree model can be used for representing any probability distribution given.

The Bayesian prototype tree model is described in more detail in Section II. Naturally the accuracy of the model depends on how well the chosen Bayesian prototypes reflect the actual probability distribution. In [22, 23] the prototypes were assumed to be provided by a domain expert. However, although using experts is viable in some domains, in many cases such prototypes need to be inferred from raw data. A probabilistic evaluation criterion for comparing different prototype tree models is represented in Section III. Using this evaluation criterion as an example, we show in Section IV how to use the simulated annealing algorithm for searching for the best model for the problem domain.

II. The Bayesian prototype tree model

We represent the problem domain knowledge by using values of m discrete attributes (random variables) $A_1, \ldots, A_m$, where each attribute A_i has n_i possible values, $a_{i1}, \ldots, a_{in_i}$. Our *training data set* $\mathcal{D}$ consists of N complete data vectors $\vec{d}_1, \ldots, \vec{d}_N$,

$$\vec{d}_h = (V_h(A_1), \ldots, V_h(A_m)),$$

where $V_h(A_i)$ is the value of attribute A_i,

$$V_h(A_i) \in \{a_{i1}, \ldots, a_{in_i}\}.$$

Incomplete data vectors are transformed to complete vectors by the theoretically correct technique of adding an 'unknown' category to the value sets of the attributes with missing values [4].

Let us cluster the N data vectors of the training set into l classes $\mathcal{C}_1, \ldots, \mathcal{C}_l$, and provide each data vector $\vec{d}_h$ with the corresponding class index $c(h) \in \{1, \ldots, l\}$. We regard the class indexes as mutually exclusive values of a hidden random variable, $\mathcal{C}$, and model each class $\mathcal{C}_k$ by a *Bayesian prototype vector*

$$\begin{aligned}\vec{c}_k &= (P_k(a_{11}), \ldots, P_k(a_{1n_1}), \ldots, \\ &\quad P_k(a_{m1}), \ldots, P_k(a_{mn_m})),\end{aligned}$$

where $P_k(a_{ij})$ expresses the likelihood for attribute A_i to have value a_{ij} in class $\mathcal{C}_k$:

$$P_k(a_{ij}) = P(A_i = a_{ij} | \mathcal{C} = k).$$

These conditional probabilities can be either estimated by simply using the relative frequencies of value occurrences, or as noted in [8, 14], approximated more accurately with the formula

$$P_k(a_{ij}) = \frac{F_k(a_{ij}) + 1}{|\mathcal{C}_k| + n_i}, \qquad (1)$$

where $F_k(a_{ij})$ is the number of vectors in cluster $\mathcal{C}_k$ with the property $A_i = a_{ij}$, and $|\mathcal{C}_k|$ is the total number of vectors partitioned in cluster $\mathcal{C}_k$.

If the cluster structure chosen conforms to the true probability distribution of the problem domain (in other words, the prototype vectors $\vec{c}_k$ represent the distributions within each cluster well), knowing membership in a cluster is sufficient information for determining the values of the attributes. In the Bayesian framework this means that *all the variables A_i are conditionally independent of each other, given the value of the variable $\mathcal{C}$* (which does not, however, mean that the attributes are independent). In other words, the Bayesian network corresponding to this representation is a tree, where a single variable $\mathcal{C}$ is the root of the tree, and variables A_i form the leaves (see Figure 1).

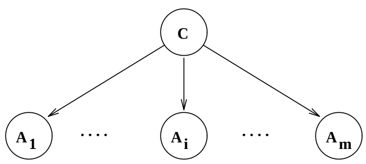

Figure 1: Bayesian prototype tree representation of the problem domain probability distribution with attributes $A_1, \ldots, A_m$.

To be able to use our Bayesin prototype tree model for Bayesian reasoning, in addition to the conditional probabilities $P_k(a_{ij})$ we need also l prior probabilities, one for each of the classes $\mathcal{C}_k$. As the conditional probabilities in (1), these can be approximated by

$$P(\mathcal{C}_k) = \frac{|\mathcal{C}_k| + 1}{N + l}. \qquad (2)$$

The accuracy of the resulting Bayesian network representation of the problem domain probability distribution depends on two factors:

1. How well does the data in the training set $\mathcal{D}$ represent the true probability distribution?
2. How well does the clustering scheme chosen reflect real clusters in the problem domain?

In this paper, we assume that the training set contains a good, representative sample of the problem domain data, and concentrate on the second problem, choosing a proper clustering scheme for the given data set. In principle our approach is akin to the work in [6], but instead of heuristic evaluation criteria we use MDL approximations and for search theoretically sound simulated annealing replacing the conceptual clustering algorithm (COBWEB). We will now proceed by showing how different clustering schemes can be compared by evaluating the corresponding Bayesian tree models.

III. An evaluation criterion for prototype trees

Let $\mathcal{M}$ denote a Bayesian prototype tree constructed as described in the previous section using a data set $\mathcal{D}$. Assuming that the training set $\mathcal{D}$ represents the problem domain distribution well, an optimal prototype tree is obtained by maximizing the probability

$$P(\mathcal{M} \mid \mathcal{D}) = \frac{P(\mathcal{M})P(\mathcal{D} \mid \mathcal{M})}{P(\mathcal{D})}.$$

As the probability $P(\mathcal{D})$ is constant for a given training set, we can compare the likelihood of different prototype sets by using as the criterion the formula

$$P(\mathcal{M} \mid \mathcal{D}) = cP(\mathcal{M})P(\mathcal{D} \mid \mathcal{M}), \qquad (3)$$

where c can be any positive constant. This criterion can also be approximated using the Minimum Description Length [26] approach, where the goal is to minimize the sum

$$-\log P(\mathcal{M}) - \log P(\mathcal{D} \mid \mathcal{M}).$$

There are many different approaches to approximating the probabilities $P(\mathcal{M})$ and $P(\mathcal{D} \mid \mathcal{M})$. For example, applying the theorems of Cooper and Herskovitz in [8], in our framework with m attributes and n_i values for each attribute A_i, we get for a Bayesian prototype tree with l prototypes the probability

$$P(\mathcal{D} \mid \mathcal{M}) = \frac{(l-1)!}{(N+l-1)!} \prod_{k=1}^{l} |\mathcal{C}_k|! \cdot \prod_{i=1}^{m} \prod_{k=1}^{l} \left(\frac{(n_i - 1)!}{(|\mathcal{C}_k| + n_i - 1)!} \prod_{j=1}^{n_i} F_k(a_{ij})! \right). \qquad (4)$$

The parameters needed for applying this formula can be computed by using the partition of the data $\mathcal{D}$ used for creating the prototype set corresponding to the tree model $\mathcal{M}$. An alternative formula for computing $P(\mathcal{D} \mid \mathcal{M})$ is given in [16]. In [8], Cooper and Herskovitz did not take into the account the prior probabilities $P(\mathcal{M})$, but assumed all structures to be equally probable. In the following, we use the size of the prototype tree for determining the prior probabilities, introducing a bias towards simple structures.

To approximate the prior probability $P(\mathcal{M})$, let us first code all the different Bayesian trees using the information theoretically optimal coding scheme in [26]. Using this coding scheme, we can approximate the prior probability as

$$P(\mathcal{M}) = 2^{-S(\mathcal{M})},$$

where $S(\mathcal{M})$ denotes the number of bits needed to represent Bayesian prototype tree $\mathcal{M}$. Our model contains l prototype vectors and l prior probabilities, one for each class $\mathcal{C}_k$. In addition, to use our Bayesian prototype tree for Bayesian reasoning, we need one prototype vector for representing the initial attribute-value distribution and the outcome of the computation. Hence the total description length for the Bayesian tree structure is

$$S(\mathcal{M}) = l\sum_{i=1}^{m} n_i r + lr + \sum_{i=1}^{m} n_i r, \qquad (5)$$

where r is the number of bits needed to store a truncated real value. Observe that for a particular problem domain m, n_i and r will be constants, but l can vary from 1 to N (the size of the data set). This approach is very similar to the MDL approximation of $S(\mathcal{M})$ in [16], but as their model is used for learning general multiply connected Bayesian networks, the encoding scheme is biased towards sparse networks with few values, and in our case it will give too high a penalty for introducing new values to the root node, i.e., for introducing new classes (class values for $\mathcal{C}$). This is not the only encoding scheme possible, but it is straightforward and reflects well the intuitive bias towards "simpler" trees, i.e., trees with the root node having less possible values.

Combining the results above, we get

$$P(\mathcal{M} \mid \mathcal{D}) \propto 2^{-S(\mathcal{M})} P(\mathcal{D} \mid \mathcal{M}), \qquad (6)$$

where $S(\mathcal{M})$ is the size of the Bayesian tree structure as defined in (5), and $P(\mathcal{D} \mid \mathcal{M})$ is computed using formula (4). Observe that in our Bayesian framework the evaluation criterion and the actual search algorithm used for finding the most probable model are completely independent of each other, and hence the simulated annealing approach described in the next section can be used with various different evaluation criteria, not only with the one example given above.

IV. Searching the prototype tree model space by simulated annealing

Given a Bayesian tree with l prototypes, and a partition of the data $\mathcal{D}$ into these l classes, we can use the criterion (6) for computing the probability of the given Bayesian tree structure, given the data, and hence evaluate the goodness of the tree representation. However, for a Bayesian tree with l prototypes, there are l^N ways to cluster the data, if the size of the data set is N, so it is clearly computationally infeasible to use the proposed scheme for evaluating all the possible tree structures with all the possible data partitions. In [24] we presented a greedy heuristic for the search process. This heuristic was a "bottom-up" approach which starts by partitioning all the data vectors in separate clusters, and continues by combining the two prototypes which produce the greatest increase in the probability $P(\mathcal{M} \mid \mathcal{D})$. In the following, we show how the simulated annealing algorithm can be used for performing a more elaborate stochastic local search in the model space. Our approach is supported by the results of applying simulated annealing for solving optimal clustering problems (see e.g., [3, 28]).

We start by assuming that the number of clusters to be used does not exceed N, the number of items in the training set. This is usually a safe assumption to make, since otherwise it can be argued that the size of the training set is too small for successful construction of a problem domain model. Under this assumption, we can use a *partition vector* $(c(1), \ldots, c(N))$, consisting of values of all the N data cluster indexes with $c(h) \in \{1, \ldots, N\}$, to represent a possible partitioning of the data into at most N clusters, where a cluster $\mathcal{C}_k$ is empty if $c(h) \neq k$, for all $h = 1, \ldots, N$. As in our framework a given data partition fully determines (through formulas (1) and (2)) the corresponding Bayesian prototype tree model, we can identify each model with the corresponding partition vector, and represent each $\mathcal{M}$ as a vector, $\mathcal{M} = (c(1), \ldots, c(N))$. Hence to find an optimal Bayesian prototype tree model we need to find the optimal partition vector among the N^N different vectors. In simulated annealing, this optimization task is solved by introducing a stochastic Markov chain process $\{\mathcal{M}_t \mid t = 0, 1, \ldots\}$, where X is a random variable the values of which are the partition vectors, and $\mathcal{M}_t$ is the value of X at time t. In the following, we show how to construct a process which converges to the optimal model $\mathcal{M}_*$:

$$\lim_{t \to \infty} P(\mathcal{M}_t = \mathcal{M}_*) = 1. \qquad (7)$$

Let $\mathcal{M}_t = (c_t(1), \ldots, c_t(N))$ be the the value of random variable X at time t, and let $P(\mathcal{M}_t \mid \mathcal{D})$ denote the probability $P(X = \mathcal{M}_t \mid \mathcal{D})$. In our simulated annealing process, we generate a candidate $\mathcal{M}_{t+1}$ for the next value of X by changing the cluster index $c(h)$ of one randomly chosen data vector $\vec{d}_h$ to a randomly chosen number in the range

$1, \ldots, N$. The resulting partition vector $\mathcal{M}_{t+1}$ is accepted as the next value for the random variable X with probability

$$p_t(\mathcal{M}_{t+1}) = \frac{1}{1 + (\frac{P(\mathcal{M}_t|\mathcal{D})}{P(\mathcal{M}_{t+1}|\mathcal{D})})^{\frac{1}{T(t)}}}, \qquad (8)$$

where $T(t)$ is a monotonicly decreasing function converging to zero as t approaches infinity, and the conditional probabilities $P(\mathcal{M}_t \mid \mathcal{D})$ and $P(\mathcal{M}_{t+1} \mid \mathcal{D})$ are computed using equation (6). The value of the function T is usually called the (computational) *temperature* of the process. Without the temperature parameter, the probability p_t is identical to the acceptance probability proposed by Barker in [2]. An alternative form for the acceptance probability, proposed originally by Metropolis et al. in [18], is given by

$$p_t(\mathcal{M}_{t+1}) = \begin{cases} 1 & , \text{ if } \frac{P(\mathcal{M}_{t+1}|\mathcal{D})}{P(\mathcal{M}_t|\mathcal{D})} \geq 1, \\ (\frac{P(\mathcal{M}_{t+1}|\mathcal{D})}{P(\mathcal{M}_t|\mathcal{D})})^{\frac{1}{T(t)}} & , \text{ if } \frac{P(\mathcal{M}_{t+1}|\mathcal{D})}{P(\mathcal{M}_t|\mathcal{D})} < 1. \end{cases} \qquad (9)$$

It has been argued that in practice the Metropolis form should be preferred since in the case of equally probable states, Barker's method changes the state with probability 1/2, whereas Metropolis' method changes the state with probability 1, thus offering possibly a better sampling of the states. However, it should be noted that the stochastic process generated by using the Barker's formula can be implemented extremely efficiently by using a massively parallel architecture [20, 21]. Theoretically there is no preference for either of these models, since using the candidate generation scheme described above both methods lead to a stochastic process fulfilling the condition (7) (see [20]).

Algorithm 1

1. /* start with a random clustering: */
   for h = 1 to N do
   (a) c(h) = RANDINT(1,N);
2. Construct a prototype tree model $\mathcal{M}_0$ corresponding to the initial clustering;
3. for t = 0 to ∞ do
   (a) Compute the new computational temperature $T(t)$.
   (b) If $T(t)$ is 0 then stop;
   (c) /* choose randomly a data item: */
   h = RANDINT(1,N);
   (d) /* save the current cluster index: */
   c = c(h);
   (e) /* choose randomly a new cluster: */
   c(h) = RANDINT(1,N);
   (f) Construct a new prototype tree model $\mathcal{M}_{t+1}$ corresponding to the new clustering;
   (g) Compute the probability $p_t(\mathcal{M}_{t+1})$ by using equation (8) or (9);
   (h) If ($p_t(\mathcal{M}_{t+1})$ < RANDREAL(0,1)) then c(h) = c; /* reject new clustering, restore old value */

If the computational temperature $T(t)$ is decreased slowly enough, the simulated annealing algorithm given will converge to the maximum probability clustering resulting to maximum probability model $\mathcal{M}_*$ [1]. Unfortunately, finding a proper cooling schedule for decreasing the temperature may be difficult in practice [20].

V. Conclusion

We have presented a method for deriving probabilistic Bayesian network models from data. In this work, we have restricted ourselves to a class of simple tree-structured Bayesian prototype tree models, although in principle the approach presented can be adapted also for learning multi-connected Bayesian networks. The reasons for choosing the family of Bayesian prototype tree models was twofold: firstly, the resulting Bayesian network models can be implemented extremely efficiently, especially if massively parallel hardware is available. Secondly, the prototype tree structure has an intuitively appealing, clear semantic interpretation as a memory-base reasoning system. The learning algorithm introduced is based on a stochastic simulated annealing search algorithm. We are currently experimenting with the proposed simulated annealing learning algorithm for Bayesian prototype trees, and in the future plan to apply the algorithm for well-known benchmark problems in order to compare our approach with alternative methods.

References

[1] Aarts, E., and Korst, J., *Simulated Annealing and Boltzmann Machines: A Stochastic Approach to Combinatorial Optimization and Neural Computing.* John Wiley & Sons, Chichester, 1989.

[2] Barker, A.A., Monte Carlo calculations of the radial distribution functions for a proton-electron plasma. *Aust. J. Phys.* 18 (1965), 119–133.

[3] Brown, D.E., and Huntley, C.L., A practical application of simulated annealing to clustering. *Pattern Recognition*, 25 (1992) 4, 401–412.

[4] Cheeseman, P., Kelly, J., Self, M., Stutz, J., Taylor, W., and Freeman, D., AutoClass: A Bayesian Classification System. Pp. 54–64 in

Proc. of the Fifth International Conference on Machine Learning, Ann Arbor, June 12–14, 1988 (Morgan Kaufmann).

[5] Cheeseman, P., On Finding the Most Probable Model. Pp. 73–95 in Computational Models of Scientific Discovery and Theory Formation, edited by J. Shrager and P. Langley. Morgan Kaufmann Publishers, 1990.

[6] Connolly, D., Constructing hidden variables in Bayesian networks via conceptual clustering. Pp. 65–72 in *Proceedings of the Tenth International Conference on Machine Learning*, Amherst, June 27–29, 1993 (Morgan Kaufmann).

[7] Cooper, G.F., The computational complexity of probabilistic inference using Bayesian belief networks. *Artificial Intelligence* 42 (1990) 2–3 (March), 393–405.

[8] Cooper, G.F., and Herskovits, E., A Bayesian Method for Induction of Probabilistic Networks from Data. *Machine Learning*, vol. 9 , pp. 309–347, 1992.

[9] Geman, S. and Geman, D., Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Trans. on Pattern Analysis and Machine Intelligence* 6 (1984), 721–741.

[10] Gyllenberg, M., Gyllenberg, H.G., Koski, T., and Schindler, J., Non-uniqueness of Numerical Taxonomic Structures. *Binary*, Vol. 5 (1993), 138–144.

[11] Hsu, W., Hsu, L.S., and Tenorio, M.F., The Clusnet Algorithm and Time Series Prediction. *International Journal of Neural Systems*, vol. 4, no. 3, pp. 247–255, September 1993.

[12] Kirkpatrick, S., Gelatt, D. and Vecchi, M.P., Optimization by simulated annealing. *Science 220* (1983) 4598 (May), 671–680.

[13] Kitano, H., Challenges of Massive Parallelism. Pp. 813–834 in *Proc. of IJCAI-93, the Thirteenth International Joint Conference on Artificial Intelligence*, Chambéry, France, August 1993 (Morgan Kaufmann).

[14] Kononenko, I., Successive Naive Bayesian Classifier. *Informatica*, vol. 17, pp. 167–174, 1993.

[15] Langley, P., Induction of Recursive Bayesian Classifiers. Pp. 153–164 in P.B. Brazdil (ed.), *Proc. of ECML-93, European Conference on Machine Learning*, Vienna, Austria, April 5–7, 1993 (Springer-Verlag).

[16] Lam W., and Bacchus, F., Using Causal Information and Local Measures to Learn Bayesian Networks. Pp. 243–250 in D. Heckerman and A. Mamdani (eds.), *Proc. of the Ninth Conference on Uncertainty in Artificial Intelligence*, Washington, D.C., July 9–11, 1993 (Morgan Kaufmann).

[17] Lee S., and Shimoji, S., BAYESNET: Bayesian Classification Network Based on Biased Random Competition using Gaussian Kernels. Pp. 1354–1359 in *Proc. of the IEEE International Conf. on Neural Networks*, San Francisco, March 1993 (IEEE Press).

[18] Metropolis, N., Rosenbluth, A.W., Rosenbluth, M.N., Teller, M.N. and Teller, E., Equations of state calculations by fast computing machines. *Journal of Chem. Phys.* 21 (1953), 1087–1092.

[19] Moody, J., and Darken, C., Fast learning in networks of locally-tuned processing units. *Neural Computation*, vol. 1, 281–294, 1989.

[20] Myllymäki, P., Bayesian Reasoning by Stochastic Neural Networks. Ph. Lic. Thesis, Report C-1993-67, Department of Computer Science, University of Helsinki, 1993.

[21] Myllymäki, P., Using Bayesian networks for incorporating probabilistic a priori knowledge into Boltzmann machines. *Proc. of SOUTHCON'94* (Orlando, March 1994), 97–102.

[22] Myllymäki, P., and Tirri, H., Bayesian Case-Based Reasoning with Neural Networks. Pp. 422–427 in *Proc. of the IEEE International Conf. on Neural Networks*, San Francisco, March 1993 (IEEE Press).

[23] Myllymäki, P., and Tirri, H., Massively Parallel Case-Based Reasoning with Probabilistic Similarity Metrics. Pp. 48–53 in M.M. Richter, S. Wess, K.-D. Althoff and F. Maurer (eds.), *Proc. of the First European Workshop on Case-Based Reasoning*, University of Kaiserslautern, 1–5 November, 1993. SEKI Report SR-93-12 (SFB 314), University of Kaiserslautern.

[24] Myllymäki, P., and Tirri, H., Learning in neural networks with Bayesian prototypes. *Proc. of SOUTHCON'94* (Orlando, March 1994), 60–64.

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

[26] Rissanen, J., *Stochastic Complexity in Statistical Inquiry*. World Scientific, 1989.

[27] Robinson, W.R., Counting unlabeled acyclic digraphs. In C.H.C. Little (ed.), *Lecture notes in mathematics 622: Combinatorial mathematics*. Springer-Verlag, 1977.

[28] Selim, S.Z., and Alsultan, K., A simulated annealing algorithm for the clustering problem. *Pattern Recognition* 24 (1991) 10, 1003–1008.

[29] Specht, D.F., Probabilistic Neural Networks. *Neural Networks*, vol. 3, pp. 109–118, 1990.