Proceedings of the IEEE World Congress on Computational Intelligence (Orlando, June 1994), 1493–1497.

MDL Learning of Probabilistic Neural Networks for Discrete Problem Domains

Henry Tirri and Petri Myllymäki

***Abstract*— Given a problem, a case-based reasoning (CBR) system will search its case memory and use the stored cases to find the solution, possibly modifying retrieved cases to adapt to the required input specifications. In discrete domains CBR reasoning can be based on a rigorous Bayesian probability propagation algorithm. Such a Bayesian CBR system can be implemented as a probabilistic feedforward neural network with one of the layers representing the cases. In this paper we introduce a Minimum Description Length (MDL) based learning algorithm to obtain the proper network structure with the associated conditional probabilities. This algorithm together with the resulting neural network implementation provide a massively parallel architecture for solving the efficiency bottleneck in case-based reasoning.**

I. Introduction

Our research is motivated by combining the theory of case-based reasoning (CBR) with neural network techniques, especially for developing massively parallel implementations for "shallow knowledge" domains such as the ones discussed in [4]. In case-based reasoning paradigm the dynamic case memory is central to the reasoning process and learning is an inherent part of the process [5]. Although the idea of using a case memory is simple in principle, there are many open issues related to case indexing, similarity metric in case matching and case adaptation. For example much of the published work has concentrated on applying machine learning methods for indexing cases in order to avoid costly comparison to the case set. However, as recent work demonstrates [10], one is able to use Bayesian reasoning framework [13, 11] to develop theoretically sound solutions for the CBR reasoning process.

On the other hand neural network architectures are constructed from a large amount of highly con-

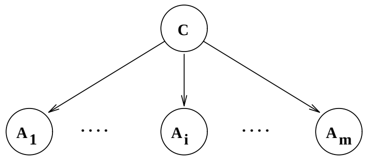

Figure 1: Bayesian network representation of the case base.

nected elements. This makes them suitable for distortion tolerant storing of a large number of cases (represented by high dimensional vectors) by making single elements "sensitive" to a stored item. In addition, as there is nothing that resembles a shared memory, the neural computing architecture is inherently parallel, and each element can perform comparison of its input against the value stored in the interconnections independently from the others. This offers a linear speed-up in the comparison process with respect to the number of computational elements available. This means that for example the case indexing problem [5] can be addressed directly at the architectural level as matching can be performed by using the available parallelism. Consequently for discrete domains Bayesian CBR reasoning implemented on a well-matching neural architecture will complement each other and providing benefits of both approaches.

In [9] we presented a neural network architecture for case-based reasoning by adopting Pearl's probability propagation calculation and represented it as a multi-layer feedforward probabilistic network. This approach has the advantages discussed above: efficient case matching inherent in the parallel architecture, theoretically sound Bayesian interpretation of the case-space metrics and one possible solution to case adaptation via similar probability propagation as the one used for case matching. This network is related to the probabilistic networks of Specht [19, 20], and other kernel-based estimators [12, 1], which assume that the domain probability distribution can be approximated as a finite mixture of certain type of kernel distributions (e.g., Gaussian). In

The authors are with the Department of Computer Science, P.O.Box 26, FIN-00014 University of Helsinki, Finland
This research was supported by the Technology Development Center (TEKES)

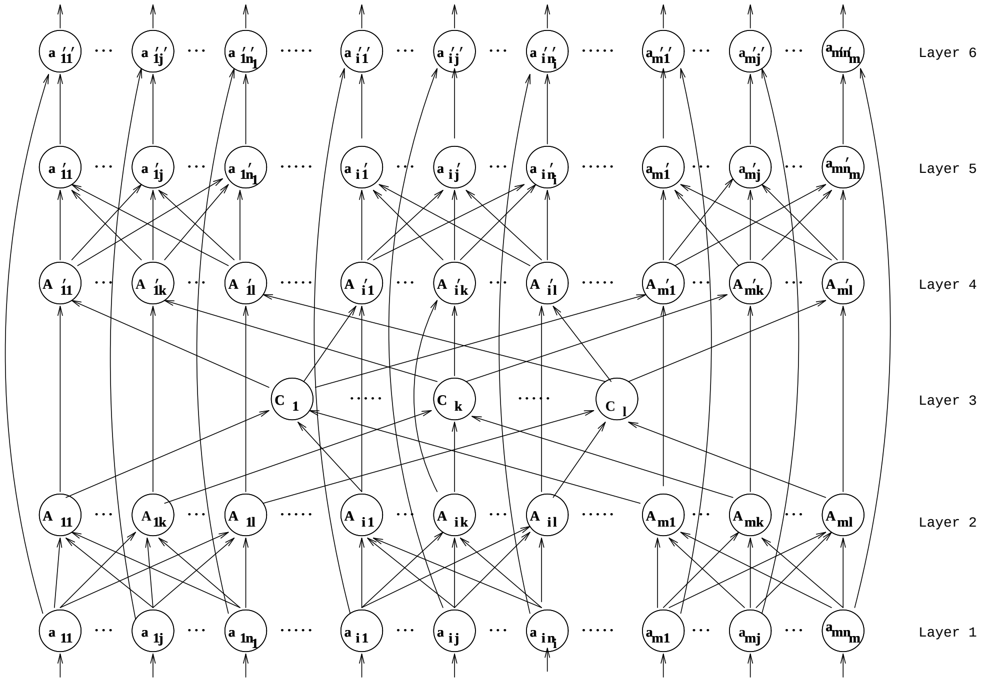

Figure 2: The probabilistic neural network implementing Bayesian CBR.

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 approach resembles the simple Bayesian classifier in [6] and [7], but unlike these models, our network can be used also for general regression tasks (case adaptation), and not only for classification tasks (case matching). In principle the simple Bayesian classifier model corresponds to the first three layers of the network shown in Fig. 2.

In [9], we left open the problem of learning, i.e., choosing the cases (kernels) in the case layer (layer 3 in Fig. 2) of the architecture[1], and determining the conditional probabilities needed for the model. In this paper we will address this problem and present a Minimum Description Length [16] learning framework for selecting a proper case set together with associated conditional probabilities. It should be observed that as in the general case the proper number of cases (actually clusters) is not known in advance, the learning framework is unsupervised in nature. We proceed first in Section 2 by defining the learning problem and then briefly describing the neural network architecture used and the corresponding underlying Bayesian case-based reasoning network structure. This Bayesian CBR structure allows us to develop the MDL based selection criteria for learning analogous to the one presented in [8] (see Section 3). In Section 4 we will outline the learning algorithm, and Section 5 gives the conclusions.

[1] One should observe that in noisy domains the best strategy is not necessary to store all the cases encountered

II. The learning problem

Let us assume that the problem domain knowledge is represented using values of m discrete attributes (random variables) $A_1, \ldots, A_m$, where each attribute A_j has n_j possible values, $a_{j1}, \ldots, a_{jn_j}$. Our *training data set* consists of N data vectors $\vec{d_i}$,

$$\vec{d_i} = (V_i(A_1), \ldots, V_i(A_m)),$$

where $V_i(A_j)$ is the value of the attribute A_j, $V_i(A_j) \in \{a_{j1}, \ldots, a_{jn_j}\}$. Generally, the data vectors are not uniformly distributed over the vector space, but form clusters of one or more vectors. Let us cluster the N data vectors of the training set into l classes ($l \leq N$), and provide each data vector with the corresponding class label. If the cluster structure chosen corresponds well to the true probability distribution of the problem domain, knowing membership in a cluster implies good estimates for the attribute values.

The class labels $c_1, \ldots, c_l$ can be regarded as mutually exclusive values of a hidden random variable, C. In the Bayesian framework this means that *all the variables A_i are conditionally independent of each other, given the value of the variable C* (however, this does not mean that the attributes are independent). In other words, the Bayesian network corresponding to this representation is a tree, where a single variable C is the root of the tree, and variables A_i form the leaves (see Fig. 1). This simple structure allows us to adopt the probability propagation algorithm developed by Pearl [13] and implement it as a probabilistic feedforward network presented in Fig. 2. The details of the propagation algorithm and the neural network structure are presented in [9]. However, to construct such networks one needs a learning algorithm to determine the arc weights, i.e., the conditional probabilities $P(A_i = a_{ij}|C = c_k)$ and the structure of the network. The conditional probabilities can be estimated from the relative frequencies in the training data, when the structure of the network is fixed and the training data is labeled using the case labels chosen. Learning the structure means for our class of networks determining the proper number of cases, which is not known in advance but should be calculated based on the training data. A criteria for evaluating different structures based on the MDL principle is described in the following section.

III. MDL criteria for learning

In an abstract setting, learning using MDL principle is based on the idea that the best network representing the data set is the network that minimizes the sum consisting the length of encoding of the network and the length of encoding the data given the network [16]. Thus applicability of the MDL principle depends on the proper encoding of both of these components. Finding such encodings for feedforward neural network models has proven to be a non-trivial task (see e.g., discussions in [18]). Since in our case we have a tight correspondence between the Bayesian tree and the probabilistic network, we are able to make use of the simpler structure and develop the MDL criteria at the Bayesian tree structure level. Hence we will now illustrate how such Bayesian tree structures can be learned, the resulting tree can then be trivially transformed into the respective neural network.

Work on learning Bayesian networks has recently increased as the networks have become more widely applied. Of the research related to ours [2, 3, 14] the closest to the ideas applied here is that of Lam and Bacchus [8]. They suggest an MDL learning scheme for general, multiply connected Bayesian networks. However, their encoding scheme is biased towards sparse networks with few values. In our case it will give too high a penalty for introducing new values to the root node, i.e., for introducing new cases. This observation was also verified empirically by experimenting with the criteria presented in Section 2 of [8]. For our neural network structure their criteria will usually produce only one node at the "case-layer" (layer 3 in Fig. 2).

Encoding the Bayesian tree structure Our encoding scheme for the Bayesian tree has the property that the more values the node C has, the longer will be its encoding. This is due to the fact that each arc in Fig. 1 represents *a set of conditional probabilities* $P(A_i = a_{ij}|C = c_k)$, thus each new value (i.e., case) will introduce new conditional probabilities for all attribute-value pairs. The encoding of the tree consists of three components: nodes, conditional probabilities and a priori probabilities of the case node values. Suppose there are m attribute nodes in the tree, and each attribute A_i can take on n_i values. The case node C can have l values. Then we need

$$\sum_{i=1}^{m} \log n_i + \log l$$

bits to represent the nodes[2]. Let r be the number of bits needed to store a truncated real value. Then to represent the conditional probabilities we need

$$l \sum_{i=1}^{m} n_i r$$

bits and to represent the a priori probabilities lr bits. 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). Thus the total description length for the Bayesian tree structure will be:

$$\sum_{i=1}^{m} \log n_i + \log l + l \sum_{i=1}^{m} n_i r + lr. \qquad (1)$$

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. This translates in the neural network level to a probabilistic network with fewer nodes in the case layer.

Encoding the data using the tree Giving a usable criteria for calculating the length of the encoding of the data given a Bayesian tree is more difficult. In principle to represent the data as a binary string we can use standard coding techniques. Lam and Bacchus [8] suggest the use of Huffmann coding for the data encoding, although in fact the optimal coding would be achieved by arithmetic coding [15]. However, since we are only interesting in the length of the resulting code string, the actual choice of the encoding scheme is not of much importance. Both

[2]All logarithms in this paper are of base 2.

Huffmann and arithmetic coding rely on probability estimates q_i of the data elements and assign a codeword of length $-\log q_i$ to the element $\vec{d_i}$. Since we are encoding the data using a Bayesian tree, the probability estimates q_i are produced by the particular tree under consideration. It is well known that the shortest code length will be achieved, when the probability estimates q_i are the true probabilities p_i (see e.g., [16]). Therefore a good approximation for the length of the data coded with the Bayesian tree will be

$$-N \sum_i p_i \log(q_i) \qquad (2)$$

where the sum is over all the possible data vectors. In the equation it is assumed that for large data sets we will have approximately Np_i occurrences of a data element $\vec{d_i}$. Consequently this data encoding component measures "the goodness of fit" of the learned Bayesian tree with respect to the data set. Unfortunately this component is computationally intractable and needs to be approximated as described in the next Section.

Hence the total description length to be used in learning as a selection criteria is formed by adding (1) and (2) which results in:

$$\sum_{i=1}^{m} \log n_i + \log l + l \sum_{i=1}^{m} n_i r + lr - N \sum_i p_i \log(q_i) \quad (3)$$

.

IV. Searching the solution space

As the formula (3) gives us a method for computing the description length of a given Bayesian tree structure, given a set of data, we could in principle go through all the possible structures, evaluate them using the description length criteria, and choose the optimal (MDL) solution. Although in the general case the number of possible Bayesian network structures grows super-exponentially with the number of nodes [17], in our tree structure domain it is reasonable to assume that the number of cases does not exceed the size of the data set, so one would expect to evaluate only N different tree structures. However, after introducing the latent case variable C in our network, before being able to evaluate a given Bayesian tree structure in our domain we need to complement our data set by adding values for the case variable, i.e., we need to attach to each data vector $\vec{d_i}$ some class label c_k. Obviously, there are l^N different possibilities for assigning N data vectors in l classes, so for most practical situations, we need to approximate the exact solution by using a greedy heuristic for choosing the optimal tree structure. The heuristic we propose here is a simple additive algorithm, which starts with one case, and assigns each of the data vectors $\vec{d_i}$ either to one of the already existing cases, or creates a new case, and assigns data to it:

Algorithm 1

1. $l = 1$, c = 1;
2. Assign data $\vec{d_1}$ to case c;
3. for i = 2 to N do
   (a) MDL = HUGE;
   (b) for k = 1 to l do
      i. Assign data $\vec{d_i}$ to case k;
      ii. DL = size(structure) + size(data);
      iii. If (DL $\leq$ MDL)
         A. MDL = DL;
         B. c = k;
   (c) $l = l + 1$.
   (d) Assign data $\vec{d_i}$ to case l;
   (e) DL = size(structure) + size(data);
   (f) If (DL $\leq$ MDL) go to 3;
   (g) $l = l - 1$;
   (h) Assign data $\vec{d_i}$ to case c;

The parameters size(structure) and size(data) are computed using the formulas (1) and (2), respectively.

V. Conclusions

We have presented a MDL learning procedure for probabilistic neural network architectures. Such networks can be derived from Bayesian CBR representations, and this direct correspondence allowed us to formulate the learning problem in the Bayesian network framework. Developing the scheme consisted of deriving proper encodings for both the Bayesian tree structure and the data given such a structure. These equations could then be used in the approximate learning procedure as evaluation criteria for structure selection during learning. We have implemented the probabilistic neural network architecture together with the MDL learning scheme and the initial experiments with the approach applied to medical data have produced encouraging results. However, we are also experimenting with other learning procedures such as standard clustering methods in order to be able to perform comparisons between the various approaches.

References

[1] P. Cheeseman, J. Kelly, M. Self, J. Stutz, W. Taylor and D. Freeman, 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).

[2] C. Chow and C. Liu, Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, vol. 14, pp. 462–467, 1968.

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

[4] H. Kitano, 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).

[5] J. Kolodner, **Case-Based Reasoning**. Morgan-Kaufmann, 1993.

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

[7] P. Langley, 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).

[8] W. Lam and F. Bacchus, 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).

[9] P. Myllymäki and H. Tirri, 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).

[10] P. Myllymäki and H. Tirri, 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.), *Proceedings of EWCBR-93, 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.

[11] R. Neapolitan, **Probabilistic Reasoning in Expert Systems**. Wiley Interscience, 1990.

[12] E. Parzen, On estimation of a probability distribution and mode. *Annals of Mathematical Statistics*, vol. 33, pp. 1065–1076, 1962.

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

[14] J. Pearl and T. Verma, A theory of inferred causation. Pp. 441–452 in J.A. Allen, R. Fikes and E. Sandevall (eds.), *Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning*, April 1991 (Morgan Kaufmann).

[15] J. Rissanen, Generalized Kraft-Inequality and Arithmetic Coding. *IBM J. Res. Devel.*, vol. 20, pp. 198–203, 1976.

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

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

[18] P. Smyth, On Stochastic Complexity and Admissible Models for Neural Network Classifiers. Pp. 818–824 in Lippmann, Moody, Touretzky (eds.), **Advances in Neural Information Processing Systems 3**. Morgan Kaufmann, 1991.

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

[20] D.F. Specht, A General Regression Neural Network. *IEEE Transactions on Neural Networks*, vol. 2, No. 6, pp. 568–576, November 1991.