Decision-Tree Based Neural Network (Extended Abstract)

Irena IVANOVA[1] and Miroslav KUBAT[2]

1 Department of Medical Informatics, Graz University of Technology, Brockmanngasse 41, A-8010 Graz, Austria—on leave from the Bulgarian Academy of Sciences, Institute for Information Technologies, Acad. G. Bonchev Str., Bl. 29A, 1113 Sofia, Bulgaria, e-mail: irena@iinf.bg

2 Institute of Systems Sciences, Johannes Kepler University, Altenbergerstr.69, A-4040 Linz, Austria, e-mail: mirek@cast.uni-linz.ac.at

1 Introduction: The Algorithm TBNN

The system TBNN (Tree-Based Neural Net) maps decision trees to neural networks trained by the backpropagation algorithm. This idea is not new. In Sethi (1990), the tree is mapped to a network with 2 hidden layers. Park (1994) shows a mechanism for mapping multivariate decision trees to networks with one or two hidden layers. TBNN differs from its predecessors in 1) *fully interconnecting* all adjacent layers, 2) redescribing the examples in terms of interval-membership functions, and then 3) using its own mechanism to *fuzzify these functions.*

The algorithm of TBNN is summarized in Table 1. For the induction of the decision tree we re-programmed the system ID3 (Quinlan, 1986) and used for the treatment of numeric attributes the binarization technique suggested by Fayyad and Irani (1992). The decision tree encodes one DNF for each class: the conditions along a branch are conjuncted and the individual branches are disjuncted. In the network, each branch is mapped to one hidden neuron and each class is mapped to one output neuron. Input neurons represent intervals imposed on the attributes by the tests in the internal nodes of the tree. For instance, if the tests on a_1 appeared in two places as $a_1 < 0.3$ and $a_1 < 0.7$, then there will be 3 input neurons representing the intervals $[0; 0.3), [0.3; 0.7)$, and $[0.7; 1]$ (provided a_1 is normalized to the interval $[0, 1]$).

At this point, the network is partially connected—it contains only *principal links* determined by the logic of the decision tree. The weights along the principal links are set only after the supplementation of *additional links* (with very small random weights) that make the network fully connected. The weights along the principal links are calculated from the additional weights and from the logic encoded in the decision tree so that the network closely emulates the classification behavior of the tree (the necessary formulae and their derivation can be found in Ivanova and Kubat, 1995).

To *soften* the interval-membership function, TBNN first determines the closeness of the attribute value to the interval center: $C_i = (R_i - 2\,|\,\mu_i - x_j\,|)/(2R_i)$, where μ_i is the center of the i-th interval, R_i is the size of the interval, and x_j is the actual value of the related attribute. Closeness C_i is then subjected to the sigmoid function $m_i = 1/(1 + e^{-hC_i})$ of the input neurons.

Table 1: Algorithm of the system TBNN

1. Take a *subset* of the examples and induce from them a decision tree;
2. Transform attribute values into interval-membership functions, and thus define input neurons. Turn the tree into a neural network with only those links (referred to as *principal*) that were derived from the tree;
3. Fully interconnect the adjacent layers (the new links are referred to as *additional*). Assign to the new links small initial weights;
4. Calculate the weights along the principal links so that the network emulates the decision-tree classifications;
5. Slightly perturb all weights and train the network by the backpropagation algorithm using *all* training examples.

Table 2: Classification accuracy achieved on the glass, diabetes, and breast-cancer data.

file	C4.5	C4.5-rules	LVQ	ID3	TBNN
glass	64.2 ± 3.3	62.1 ± 1.8	62.4 ± 5.3	60.9 ± 5.6	70.0 ± 2.4
diabetes	87.6 ± 3.1	87.6 ± 3.1	95.1 ± 2.1	89.0 ± 3.0	94.1 ± 2.5
cancer	94.0 ± 0.7	94.8 ± 0.6	84.5 ± 2.4	94.6 ± 0.7	96.2 ± 0.5

2 Experiments

• To test the overall performance of TBNN, we ran the program on three benchmark files representing examples by numeric attributes: *glass*, *diabetes*, and *breast-cancer* data. Missing values in the breast-cancer file were filled with the most frequent values. For performance estimation we applied the random-subsampling strategy for 10 different training/testing splits. Table 2 shows the results achieved by TBNN and compares them to some more traditional systems.

• The original idea of the algorithm was that a *subset* S_I of examples would be used for the tree induction and then a superset $S_T \supseteq S_I$ (*all* training examples) would be used for the backpropagation training. The next experiment shows to what extent the size of S_I (by fixed S_T) affects the quality of the network after the tree-to-net mapping. The results shown in Figure 1 confirm the intuition that the more examples are used for the tree induction, the better the starting point of the backpropagation algorithm.

• TBNN's potential to attack a really difficult medical task (automatic sleep classification) was tested on three subjects, BR, KR, and RA, with file sizes 920, 920, and 770 examples, respectively. Examples were described by 15 numeric attributes and categorized into 7 possible classes. There were no missing values but attributes were somewhat noisy and the classifications were inconsistent. To prevent overtraining, the backpropagation part of TBNN used a subset of the training examples as the 'training test set'—the training stopped when the accuracy on this set leveled off.

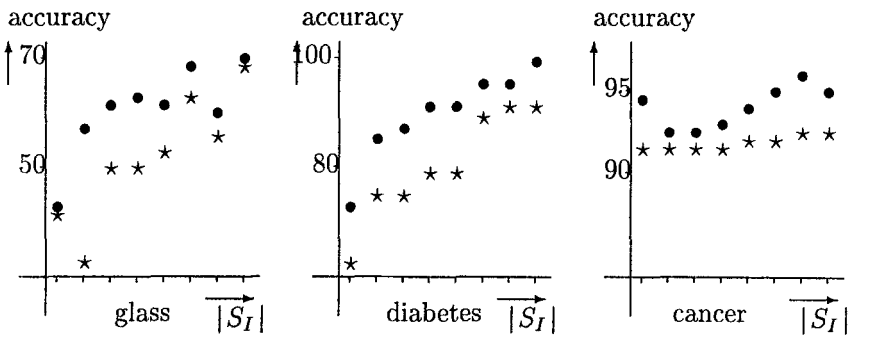

Fig.1 Effect of the size of the set S_I of examples used for generation of the tree on the performance of ID3 ($\star$) and TBNN ($\bullet$)

Table 3: Classification accuracy achieved by various systems on the sleep data. All systems were run on the same training examples (600 in the case of BR and KR, and 450 in the case of RA) and tested on the same 320 testing examples.

file	MLP	C4.5	LVQ	ID3	TBNN
BR	60.0 ± 2.8	78.1 ± 1.8	81.0 ± 1.5	77.3 ± 3.0	85.5 ± 3.6
KR	39.7 ± 5.3	61.7 ± 3.8	66.5 ± 2.5	64.0 ± 1.6	68.3 ± 4.9
RA	68.3 ± 11.0	76.8 ± 1.9	78.5 ± 2.5	77.5 ± 1.5	80.8 ± 0.6

Table 3 summarizes the classification accuracies achieved on these data files by various learners. MLP used 10 hidden units and the same overtraining-prevention strategy as TBNN. The positions of the code vectors in LVQ were determined by the k-means algorithm with $k = 2$. TBNN consistently provided the best classification accuracy. Moreover, its computational requirements (several epochs) were much lower than those of MLP (thousands of epochs).

Table 4 shows the impact of the individual aspects of TBNN: the backpropagation algorithm (B), additional links (A), and the input fuzzification (F). The dashes in the headings of columns 3 and 4 mean that the same results were ob-

Table 4: Performance of various configurations of TBNN on the sleep data

	F	F	not F	not F	not F $\star$	not F	F
	A	not A	not A	A	not A $\star$	A	A
	not B	—	—	not B	not B $\star$	B	B
BR	67.0	67.0	71.6	72.4	72.0	77.0	85.5
KR	47.6	49.3	51.5	51.7	55.6	60.1	68.3
RA	46.5	48.0	63.8	64.3	71.2	74.1	80.8

tained no matter whether the backpropagation algorithm was used or not. The column with stars in the heading contains the performance of the network with the step functions instead of sigmoids. These results are equal to those of ID3 run on S_I. In all other cases, the neurons have sigmoid functions.

3 Discussion

Neural networks have strong potential for finding representations of 'difficult' concepts thanks to the dimensionality of their search space. On the other hand, they suffer from high computational demands, local minima, saddle points, the danger of overtraining, network paralysis, strong reliance on the architecture, and the need for many training examples.

One possibility how to reduce the impact of these shortcomings is to begin with some initial approximation of the concept, provided by mapping a decision tree on a network architecture. The network training then starts from a point that is already close to the global minimum or, at least, to some good local minimum. This dramatically reduces computational requirements and helps avoiding local minima.

The research reported here studied the properties of the system TBNN that builds on these conjectures. It turns out that mere backpropagation algorithm is not enough. Only fully connected network with softened inputs will provide maximum performance.

TBNN offers the following advantages: high classification accuracy, ability do achieve high performance with modest number of learning examples, and acceptable computational requirements.

Acknowledgement

The medical data (sleep classification) used in the case study had been recorded and classified under a grant sponsored by the agency 'Fonds zur Förderung der wissenschaftlichen Forschung' (Project *S*49/03).

References

U.M. Fayyad and K.B. Irani (1992). On the Handling of Continuous-Valued Attributes in Decision-Tree Generation. *Machine Learning*, 8:87–102

I. Ivanova and M. Kubat (1995). Initialization of Neural Networks by Means of Decision Trees. *Knowledge Based Systems* 8(4), in press

Y. Park (1994). A Mapping from Linear Tree Classifiers to Neural Net Classifiers. *Proceedings of IEEE International Conference on Neural Networks*, Orlando, Florida, June–July, Vol. I, 94–100

J.R. Quinlan (1986). Induction of Decision Trees. *Machine Learning*, 1:81–106

I.K. Sethi (1990). Entropy Nets: From Decision Trees to Neural Networks. *Proceedings of the IEEE* 78:1605–1613