SELF-ORGANIZING NEURAL TREE NETWORKS

Diego H. Milone, José C. Sáez, Gonzalo Simón, Hugo L. Rufiner.
Laboratorio de Cibernética, Departamento de Bioingeniería, Facultad de Ingeniería, Universidad Nacional de Entre Ríos, Ruta 11 Km. 10, Oro Verde (CP: 3101), Entre Ríos, Argentina.
E-mail: cyberlab@fi.uner.edu.ar

***Abstract*— Automatic pattern classification is a very important field of artificial intelligence. For these kind of tasks different techniques have been used. In this work a combination of decision trees and self-organizing neural networks is presented as an alternative to attack the problem. For the construction of these trees growth processes are applied. In these processes, the evaluation of classification efficiency of one or several nodes in different configurations is necessary in order to take decisions to optimize the structure and performance of the self-organizing neural tree net. In order to perform this task a group of coefficients that quantify the efficiency is defined and a growth algorithm based on these coefficients is developed. In the tests, a comparison with other classification methods, using cross-validation methods with real and artificial databases, is carried out.**

***Index Terms*— Decision Trees, Self-Organizing Neural Networks, Automatic Classification, Neural Tree Networks.**

I. INTRODUCTION

PATTERN classification is an important tool used in experimental data analysis, automatic recognition, computer vision and other engineering and scientific disciplines. Machine-learning algorithms constitute a part of what is called automatic classification and their purpose is to extract relevant information from a set of patterns to achieve their classification. These algorithms can be either supervised or unsupervised. From another point of view they can be grouped in simple and hierarchical [1].

Decision Trees (DTs) and Artificial Neural Networks (ANNs) are two techniques widely used in the implementation of classifiers. DTs generate a set of data partitions based in a hierarchical node-structure in which comparisons upon a characteristics vector component's are carried out. DTs can be either binary or n-ary, depending on the quantity of partitions made at each node. The characteristics of the node function and the size of the tree state the complexity of the final decision boundary. One of the most commonly used functions consists of a test using a threshold for each attribute, obtaining as a result the partition of the attributes space through hyper-planes parallel or orthogonal to the coordinate axes of the space of attributes. ID3 and CART are two of the most commonly used algorithms [2, 3, 4].

ANNs [5] are formed by a set of non-linear processing units highly interconnected, that process in parallel a set of data to extract information. In 1982 T. Kohonen [1, 6] introduced a self-organizing algorithm that produced organized maps in order to solve practical classification and pattern recognition problems. These maps were named as self-organizing maps (SOMs).

An alternative to overcome some limitations of DTs, consists of hybrid implementations of DTs and ANNs. In 1991 Sankar and Mammone [7] presented what they called neural tree nets (NTNs). This kind of approach allows to take advantage of hierarchical classification properties and to create more complex decision boundaries with a smaller number of nodes. The problem with the partitions generated through hyper-planes parallel or orthogonal to the coordinate axes is that a great quantity of nodes or rules is required in order to approximate the boundaries when they are complex. NTNs are DTs that use a ANN at each node to implement the decision tasks. In this way the decision taken at each node is based on more complex rules, what permits a better boundary approximation. Several ANNs have been used at the nodes, for example: simple perceptrons (SP) [7], multi-layer perceptrons (MLP) [8, 9] and SOMs [10]. The quantity of partitions produced at each node can be fixed [7, 8] or variable [9]. The NTN has the possibility of adopting a most adequate configuration for each problem when the quantity of generated classes may vary for each node.

In this work, a SOM classifiers based self-organizing neural tree network (SONTN) growth algorithm is presented. This combination allows unsupervised nets to separate the patterns according to their natural distribution and the trees hierarchic characteristics are thus exploited.

The SONTN algorithm permits to classify in the initial levels the groups of patterns that are more separated from each other (more easily separable) and to classify patterns in the final layers in a more accurate way (patterns that are more difficult to separate). One of the main problems is the decision about the number of partitions to be made at each node in the case of n-ary trees. To solve this problem, criteria based on classification coefficients were applied.

This paper was organized as follows. The Methods section describes the classification coefficients and the training algorithm for the SONTN, following with the database used

for training. Finally results are presented for all the experiments and are discussed in the next section.

II. METHODS

Classification Coefficients

Given a general classifier, let $X = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_P\}$, where $\boldsymbol{x}_i \in \Re^D$, be the set of input patterns. These input patterns correspond to *M* classes denominated input classes $C^I = \{C_1^I, C_2^I, ..., C_M^I\}$.

Input patterns can be grouped according to the classes to which they actually belong, but they can also be grouped according to the classes C_j^O in which they are separated by the classifier. These latter classes form the set of output classes $C^O = \{C_1^O, C_2^O, ..., C_N^O\}$.

An important element for the subsequent development is the input-output intersection matrix defined as:

$$N_{i,j}^{IO} = \mathrm{n}\left(C_i^I \cap C_j^O\right) \text{ where } 1 \le i \le M \text{ and } 1 \le j \le N \qquad (1)$$

where n is the cardinality operator. This matrix contains in its *i,j-th* cell the number of patterns belonging to the input class C_i^I that were classified as belonging to the output class C_j^O.

To measure the degree in which a classifier groups patterns belonging to one input class in one output class, the inter-class concentration coefficient for the input class C_i^I in the *N* output classes C_j^O is defined as:

$$cc_i = \frac{N \max_{j=1}^{N}\left(N_{i,j}^{IO}\right) - \sum_{j=1}^{N} N_{i,j}^{IO}}{(N-1)\sum_{j=1}^{N} N_{i,j}^{IO}} \qquad (2)$$

The *inter-class concentration coefficient* for a classifier is defined as the average of the cc_i weighted by the number of patterns belonging to the corresponding input class:

$$cc = \frac{\sum_{i=1}^{M} N \max_{j=1}^{N}\left(N_{i,j}^{IO}\right) - \sum_{i,j=1}^{i=M,j=N} N_{i,j}^{IO}}{(N-1)\sum_{i,j=1}^{i=M,j=N} N_{i,j}^{IO}} \qquad (3)$$

To measure the capacity of a classifier to spread patterns from different input classes in different output classes, the intra-class dispersion coefficient for the output class C_j^O in the *M* input classes C_i^I is defined as:

$$cd_j = \begin{cases} \dfrac{M \max_{i=1}^{M}\left(N_{i,j}^{IO}\right) - \sum_{i=1}^{M} N_{i,j}^{IO}}{(M-1)\sum_{i=1}^{M} N_{i,j}^{IO}} & \text{if} \quad \mathrm{n}\left(C_j^O\right) \neq 0 \\ 0 & \text{if} \quad \mathrm{n}\left(C_j^O\right) = 0 \end{cases} \qquad (4)$$

In a similar way, the *intra-class dispersion coefficient* for a classifier is defined as the average of the cd_j weighted by the number of patterns in each output class:

$$cd = \frac{\sum_{j=1}^{N} M \max_{i=1}^{M}\left(N_{i,j}^{IO}\right) - \sum_{j,i=1}^{j=N,i=M} N_{i,j}^{IO}}{(M-1)\sum_{j,i=1}^{j=N,i=M} N_{i,j}^{IO}} \qquad (5)$$

Training algorithm for the SONTN

The whole set of training patterns is initially presented to the root node. A subset of patterns hierarchically derived from the nodes of each level of the tree arrives to the nodes of the subsequent level.

Considering a particular node, it has to be decided whether a classification task should be or not carried out. Thus, two types of nodes are distinguished: *classifier nodes* and *terminal nodes*. To decide if a node should be a classifier node or a terminal node, two characteristics of input patterns set are taken into account: the degree of classes homogeneity and the total number of input patterns. For the former reason the *concentration coefficient for input patterns* is defined as:

$$pc = \frac{M \max_{i=1}^{M}\left(\mathrm{n}(C_i)\right) - \mathrm{n}(X)}{(M-1)\,\mathrm{n}(X)} \qquad (6)$$

To decide the type of node in relation to the mentioned characteristics, they are compared to two thresholds: the minimal input patterns concentration threshold (*upc*) and the minimal input patterns quantity threshold (*unX*). In this way, a node is said to be terminal either when its input patterns concentration exceeds the *upc* threshold or when its input patterns quantity is lower than the *unX* threshold.

If a classifier node is found then it must be trained. The ANN that must be trained to implement the classification task at the node is a SOM. The input dimension of this ANN is determined by the dimension of the patterns and it is the same for the whole tree. The number of output classes and the terminal nodes define the tree final topology that should be optimized.

To determine the appropriate quantity of output classes a node growth process is used, based on the *cc* and *cd* coefficients and two minimal classification capacity thresholds, *ucc* and *ucd* respectively. Initially, a configuration with two output classes (*N=2*) is adopted, then the net is trained and its classification performance is evaluated. In the case in which

sinc(*i*) Laboratory for Signals and Computational Intelligence (http://fich.unl.edu.ar/sinc)
D. H. Milone, J. C. Sáez, G. Simón & H. L. Rufiner; "Self-Organizing Neural Tree Networks"
Proceedings of the 20th Annual International Conference of the IEEE Engineering in Medicine and Biology Society. Vol. 3, pp. 1348--1351, Oct, 1998.

any threshold is not exceeded, N is increased in one and the training and test tasks are repeated. This process finishes either when both coefficients exceed their corresponding thresholds or when N reaches the maximum permitted *maxN*. In the latter case the best of all configurations between 2 and *maxN* is chosen and the training of that node is considered concluded. This algorithm of node growth is repeated for all the nodes of each tree level and it can be seen in Fig. 1.

```
For each Level of the tree
  For each Node of the level
    Terminal Node = (pc>upc) OR (N(X)<unX)
    If not(Terminal Node) then
      N = minN
      While not(Trained Node) do
        Create Node
        Train Node
        Test Node

        If N = maxN then
          Trained Node = true
          Look for Best N
          If Best N <> maxN then
            Destroy Node
            N = Best N
            Create Node
            Train Node
          Else
            Trained Node = (cc>ucc) AND (cd>ucd)

        If not(Trained Node) then
          Destroy node
          N=N+1
  Update thresholds
```

Fig. 1. Algorithm for the growth of the SONTN.

The requirements concerning concentration and dispersion vary in relation to the depth level in the classification process. In the first stages of the classification a greater exigency concerning the concentration is proposed, while the separation based on finer details is accomplished progressively in subsequent levels, where a better dispersion in the classification is required. Thus, unsupervision turns gradually into supervision and the conformity between output and input classes is progressively achieved.

Once the tree has been trained, the following step is the labeling of terminal nodes. The assigned-label choice to each node is accomplished according to the following equation:

$$J_i = \arg\left[\max_{j=1}^{N}\left\{P(x_i \in C_j^O | x_i)\right\}\right] \quad (7)$$

Terminal nodes are joined together, according to their labels, in another artificial level of nodes that possess the labels of all the classes and in this way in the SONTN as a whole $M = N$ is accomplished.

Training databases

The databases used to test the algorithm were taken mostly from [11]. In the mentioned work a comparison between several paradigms is presented. These results are used to make a comparison with that obtained in the present work.

Artificial and real databases were used. **CLOUDS** was selected among artificial databases. From real databases, **IRIS** (because it is widely known [12]) and two phoneme recognition databases (**PETERSON** [13] and **PHONEME**) were used.

III. RESULTS

In this section the final results of the validation tests for a number of well known classification problems are presented. The validation method used is *averaged hold out* [11]. To establish a point of comparison with the results obtained with the SONTN, results obtained with other classification techniques applied to the same problem and the Bayes maximum performance estimated with the *k-nearest-neighbor* [11] method are shown. Tables I through III show a comparison between recognition coefficients (*cr*) obtained with the SONTN; and *cr* from MLP, *learning vector quantization* (LVQ) and *simple perceptron trees* (SPT), extracted from the comparative work [11]. Table IV shows a comparison between results obtained with SONTN, MLP and LVQ using the **PETERSON** database.

An MLP and an SONTN were also trained with the **PETERSON** database to have a training speed reference. The final result shows that, for a similar performance, the SONTN is more than 8 times faster than the MLP.

TABLE I
COMPARATIVE RESULTS FOR THE **IRIS** DATABASE.

cr %	μ	Min	Max
MLP	95.78	93.33	98.61
LVQ	93.83	89.44	98.67
SPT	93.33	86.61	98.61
SONTN	97.78	96.67	98.33
Bayes	96.66	94.72	97.27

TABLE II
COMPARATIVE RESULTS FOR THE **CLOUDS** DATABASE.

cr %	μ	Min	Max
MLP	87.66	86.77	88.33
LVQ	87.66	85.66	88.55
SPT	85.66	85.00	86.33
SONTN	83.93	82.20	87.20
Bayes	88.11	87.44	88.77

TABLE III
COMPARATIVE RESULTS FOR THE **PHONEME** DATABASE.

cr %	μ	Min	Max
MLP	83.63	82.36	84.69
LVQ	83.00	82.20	83.76
SPT	83.47	82.04	85.06
SONTN	84.94	84.20	85.86
Bayes	87.7	86.81	88.02

TABLE IV
COMPARATIVE RESULTS FOR THE **PETERSON** DATABASE.

cr %	μ	Min	Max
MLP	83.08	79.55	84.85
LVQ	84.36	74.81	87.57
SONTN	86.36	84.85	87.88

sinc(*i*) Laboratory for Signals and Computational Intelligence (http://fich.unl.edu.ar/sinc)
D. H. Milone, J. C. Sáez, G. Simón & H. L. Rufiner; "Self-Organizing Neural Tree Networks"
Proceedings of the 20th Annual International Conference of the IEEE Engineering in Medicine and Biology Society. Vol. 3, pp. 1348--1351, Oct, 1998.

IV. Discussion

If results obtained with the SONTN are compared to other classifiers, it is observed that the comparative performance is good for all databases. The SONTN achieved the best performance with the **IRIS, Phoneme** and **Peterson** databases when compared to all other architectures. For the **Clouds** database the efficiency of the SONTN was 3.72% inferior to that obtained with MLP, which was the classifier that obtained the best result. It should be noticed that **Clouds** is the only artificial database.

A very important consideration to take into account when making comparisons between different architectures is the fact that the SONTN adapts its topology to the particular problem, while other methods like LVQ or MLP need an initial configuration to be specified, generally based on user experience and adjusted through trial and error. In fact, the results for LVQ, MLP and SPT shown in Tables I through III correspond to the best configuration found after several trials.

Since calculations required for the generation of an SONTN are simple, this architecture is considerably faster than other architectures with more complex algorithms. However, to solve the same problem, it may require more operations than the classic tree-generation methods like ID3. The main advantage of the SONTN compared to these methods is the ability to generate much more complex boundaries.

V. Conclusions

In this work a growth algorithm for self organizing neural tree networks is presented and its performance is compared to other classification methods using several databases. The main source of advantages of this method is based in the combination of different classification paradigms, what permits to take advantage of the properties of each of them. The algorithm outlined here combines advantages of supervised and unsupervised learning. For the definition and growth of the tree topology information concerning patterns identity is used. On the other hand, for the classification task at each node a SOM-based characteristics extractor, which has no patterns identity information, is used.

Another combination of classification paradigms found in this algorithm is that of simple an hierarchical classifiers. The general structure responds to hierarchical classification methods, while a typical simple classifier, such as a neural net, is used at each node. The SONTN is a very flexible classifier that does not need the *a priori* definition of the topology (as is required in the case of MLP). The speed of the method makes the SONTN to be an adequate configuration for the implementation of classification tasks. Hardware implementation simplicity is another interesting characteristic of this algorithm.

At this time new classifiers are being developed to reinforce nodes with poor classification performance. The time dependence of decision in a dynamic NTN is also being studied.

Acknowledgment

The authors would like to thank D. Zapata for many helpful discussions. The authors are also grateful to A. Sigura and D. Tochetto for their programming collaborations.

This work is supported by Universidad Nacional de Entre Ríos.

References

[1] T. Kohonen, "*The Self-Organizing Map*", New York: Springer-Verlag, 1995.

[2] L. Breiman, J.H.Friedman, R. A. Olshen, C. J. Stone, "*Classification and Regression Trees*", Wadsworth. Int. 1984.

[3] Sestito, Dillon, *"Automated Knowledge Acquisition*", Prentice Hall 1994.

[4] J.C. Goddard, F.M. Martínez, A.E. Martínez, J.M. Cornejo, H.L. Rufiner, R.C. Acevedo, *"Aprendizaje Maquinal en Medicina: Diabetes, un caso de Estudio";* Revista Mexicana de Ingeniería Biomédica vol.16, no.2, Octubre de 1995.

[5] R. P. Lippmann, "*An introduction to computing with neural nets*", IEEE ASSP Mag., vol.4, no.2, 1987, pp 4-22.

[6] T. Kohonen, "*The Self-Organizing Map*", Proc.IEEE, vol.78, no.9, Sept.1990, pp 1464-1480

[7] A. Sankar, R. J. Mammone, "*Neural Tree Networks*", en "Neural Networks: Theory and Applications", R. Mammone and Y. Zeevi Eds., New York: Academic Press, 1991, pp 281-302.

[8] I. K. Sethi, *"Decision tree performance enhancement using an artificial neural network implementation",* Artificial Neural Networks and Statitical Patter Recognition. Old and New Conecctions, Ed. I.K.Sethi, A.K.Jain, Elsevier Science Publishers B.V., 1991.

[9] M. G. Rahim, "*A self-learning neural tree network for phone recognition*", Artificial Neural Networks for Speech and Vision, Ed.R. J. Mammone, Chapman & Hall, 1993.

[10] C. Ke, Y. Xiang, CH. Huisheng, Y. Liping, "*Modular-Tree: A Self-Architecture Neural Network Architecture*", Report, Vol. 32, Nº 1 pp 110-119.

[11] Guerin-Dugue, A. and others, Deliverable R3-B4-P - Task B4: Benchmarks, Technical report, Elena-Nerves II *"Enhanced Learning for Evolutive Neural Architecture"*, ESPRIT-Basic Research Project Number 6891, June 1995.

[12] UCI Repository of Machine Learning Databases and Domain Theories (ics.uci.edu: pub/machine-learning - databases).

[13] G. E. Peterson, H. L. Barney, *"Control Methods used in a study of the vowels"*. J. Acoustic. Soc. Am. 24, 175-184, 1952.