ESANN'1994 proceedings - European Symposium on Artificial Neural Networks
Brussels (Belgium), 20-21-22 April 1994, D-Facto public., ISBN 2-9600049-1-4, pp. 141-146

Improving Piecewise Linear Separation Incremental Algorithms using Complexity Reduction Methods

J.M. Moreno*, F. Castillo, J. Cabestany

Universitat Politècnica de Catalunya
Dept. d'Enginyeria Electrònica
Mòdul C-4, Campus Nord
c/Gran Capità s/n
08034 - Barcelona - Spain

Abstract. In this paper we present a new method for improving the performance of Piecewise Linear Separation (PLS) incremental algorithms. As has been previously stated, this kind of neural incremental algorithms yield poor performances when dealing with some real world discrimination problems. This undesirable behaviour is due to the nature of the goal functions (which usually count the number of correctly classified input patterns) the unit training algorithms try to maximize. So as to avoid this poor performance, we shall present a method that modifies the unit training algorithms in order to optimize goal functions which are related to the distribution of the training patterns in the categories defined in the input space. As will be shown, this method improves the PLS incremental algorithms' behaviour, which may be measured as the average number of units generated by the network.

1. Introduction

Motivated mainly by the problems associated with classical neural algorithms (convergence properties, heuristic determination of network structure, etc.), there has been an increasing interest on the research field concerned with the so called neural evolutive algorithms in the last few years. The defining characteristic of this kind of neural algorithms is their capability to determine the most suitable network architecture needed in solving a certain task.

As was pointed out in [1], there are four main types of incremental evolutive algorithms, and we shall concentrate on the Piecewise Linear Separation (PLS) incremental algorithms. This kind of incremental algorithms, used mainly for classification tasks, try to find the discriminant function which separates categories defined in the input data space. This discriminant function is obtained by combining the linear discriminant functions associated to perceptron-like units generated during

*Holder of an FI research Grant under the Generalitat de Catalunya's Educ. Dept.

ESANN'1994 proceedings - European Symposium on Artificial Neural Networks
Brussels (Belgium), 20-21-22 April 1994, D-Facto public., ISBN 2-9600049-1-4, pp. 141-146

the training process.

In this paper we shall review the methods which have been used for training the individual units generated by the PLS incremental algorithms,and shall point out how these methods can cause the PLS algorithms to evolve in useless manner. We shall then present some of the already existing methods for avoiding this erroneous behaviour, and finally, a novel modification criterion for the unit training process will be explained.

2. Unit training principles

Perceptron [2] or Pocket [3] are the usual training algorithms employed for the units generated by the PLS incremental algorithms. When the learning process is completed, each unit has the weight vector which yields the best correct classification rate in accordance with its input training set.

However, optimizing the function which accounts for the number of patterns correctly classified may impose an erroneous evolution scheme for the incremental algorithm. Let us consider as an example the problem depicted in figure 1, which consists of finding the discriminant function which separates the "x" patterns from the "o" patterns.

In this figure we have represented with dotted lines the four possible discriminant functions given by the Perceptron or Pocket learning algorithms after the training process for the first unit has completed. Supposing that these discriminant functions classify on their right side category "x", we can then deduce that discriminant 0 classifies correctly 4 input patterns, discriminant 1 and 3 classify correctly 3 input patterns, while discriminant 2 classifies correctly 2 input patterns.

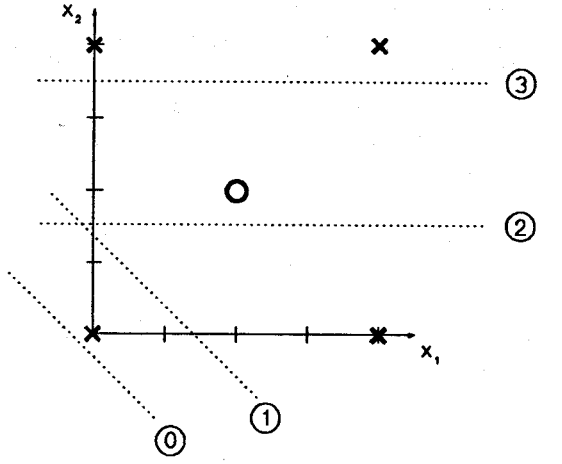

Figure 1. Possible perceptron solutions

ESANN'1994 proceedings - European Symposium on Artificial Neural Networks
Brussels (Belgium), 20-21-22 April 1994, D-Facto public., ISBN 2-9600049-1-4, pp. 141-146

Once the first unit has been trained, the PLS incremental algorithm would evolve the network by adding new units. These units receive as input training sets the reduced versions of the original training set obtained subdividing this set in as many subsets as new regions formed by the discriminant associated to the first unit. In this way, as discriminant 0 gives using the Perceptron or Pocket algorithms the best solution for the problem (it yields the best correct classification rate), the PLS incremental algorithm will produce an infinitely large network structure composed of redundant units, each of them trying to solve the same problem and arriving to the same useless solution.

There is therefore a need for improving the unit training algorithms, so as to obtain suboptimal (in the sense of correct classification rates) but useful (in terms of network evolution) solutions.

3. Modifying the unit training algorithms

Some methods have been proposed in [4], [5], [6] for improving the performance of the PLS incremental algorithms. These methods force the unit training algorithms to maximize some entropy-like function which gives information regarding the quality of the partition the current units performs on their input training set.

Furthermore, there are some other heuristic methods [7], [8] which, by modifying the unit training algorithms so as to avoid the evolution-useless solutions, provide useful network construction schemes .

The method we propose in this paper is based on the definition of some functions which provide a quantitative measure of the geometrical relations between the different categories defined in the input space.

For a problem consisting of separating M different categories, we define the category-i centroid as the vector whose components are obtained by calculating the mean value from the components of the vectors which represent the patterns belonging to this category. Once all C_i centroids have been calculated, we define the following functions for the entire distribution:

$$J_C = \sum_{p=1}^{P} \sum_{k=1}^{M} \sum_{l=1, l \neq k}^{M} J_{kl}^{p} \qquad (1)$$

$$J_{kl}^{p} = \sum_{i=1}^{N_k} || X_i^k - C_l || \qquad (2)$$

ESANN'1994 proceedings - European Symposium on Artificial Neural Networks
Brussels (Belgium), 20-21-22 April 1994, D-Facto public., ISBN 2-9600049-1-4, pp. 141-146

where:

J_{kl}^{P}: Contribution to the J_C function from the patterns belonging to the class k with respect to the centroid of class l within partition p
P: Number of partitions performed by the current unit
N_k: Number of patterns belonging to category k in partition p
x_i^k: ith pattern of category k in partition p

Figure 2 depicts, for a 3-category input problem and without partition, how some J_{kl} terms are actually calculated.

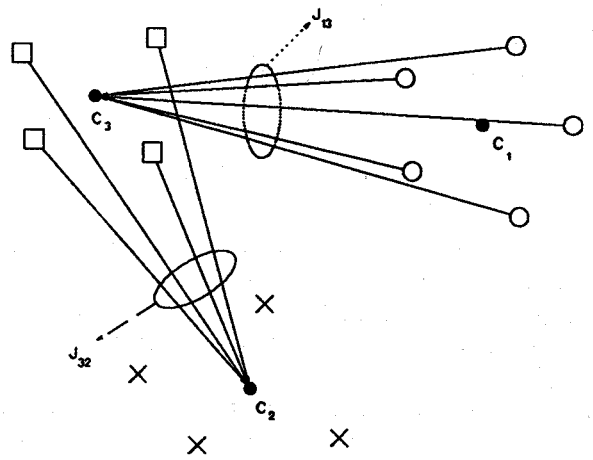

Figure 2. Calculating the value of J_C

As can be easily deduced, as the network training progresses, that is, as new partitions are formed in the input training space, the value of J_C decreases. Finally, when the global discriminant function is obtained and all the categories are correctly discriminated, J_C becomes zero, meaning that each partition contains exclusively one class.

From the previous results, we propose to modify the Pocket algorithm so as to minimize the J_C function. In this way, when the Pocket algorithm is running, we store the current weight vector only if its J_C value is lower than the last stored weight vector's J_C value.

If we now try to calculate the J_C values for the four different possible partitions obtained by perceptron like algorithms on the problem presented on figure 1 (assuming that the input patterns have components within the interval [0.0,1.0] and that manhattan distance is used), we obtain a value of 4.0 for discriminant 0, a value of 2.5 for discriminants 2 and 3, and a value of 3.33 for discriminant 1. Therefore, opposite to maximizing the correct classification rate, by choosing the partition with the lowest J_C we shall always choose a useful solution.

In table 1 we show the results, measured as mean number of units generated,

ESANN'1994 proceedings - European Symposium on Artificial Neural Networks
Brussels (Belgium), 20-21-22 April 1994, D-Facto public., ISBN 2-9600049-1-4, pp. 141-146

for the Upstart algorithm [9] using different classification problems. Each problem consists of 40 input patterns, 20 belonging to each class. For the random boolean function problem we selected 40 8-bit input patterns, for which the boolean function should produce 1 or 0 outputs with equal probability. The results are given for the Upstart algorithm [9], one of the proposed PLS incremental algorithms, with the standard Pocket algorithm and with the Pocket algorithm modified using the criterion proposed in this paper.

The results were obtained by running the Upstart algorithm 20 times on the same input problem, and then taking the mean value of the number of units generated by the algorithm in each run.

Problem	Upstart with standard Pocket	Upstart with modified Pocket
Concentric classification	24.773	9.909
Gausian classification	9.727	5.636
Intertwined spirals	116	7.227
Random boolean function	25.273	8.318

Table 1. Comparative results for the Upstart algorithm

As can be deduced from these results, the performance of the Upstart algorithm (measured as mean number of units generated) is slightly better when using the criterion proposed in this paper for training the units generated during the network construction process.

4. Conclusions

We have shown in this paper that the linear discriminant solutions given by perceptron or Pocket learning algorithms are not always useful for evolving the network structures generated by PLS incremental algorithms.

After reviewing some modification proposals for these unit training algorithms, we presented a novel approach for improving the quality of the solutions given by these algorithms. This method is based on defining some functions which measure the complexity of the distributions to be discriminated. These algorithms are modified in order to minimize the complexity measure given by those functions, so as to improve the linear discriminant functions given by the unit training algorithms,.

ESANN'1994 proceedings - European Symposium on Artificial Neural Networks
Brussels (Belgium), 20-21-22 April 1994, D-Facto public., ISBN 2-9600049-1-4, pp. 141-146

As a result of using these modified training algorithms, the performance of the PLS incremental algorithms improves significantly.

Acknowledgments

This work has been partially funded by ESPRIT III Project ELENA-Nerves 2 (no. 6891) and spanish CICYT action TIC92-629.

References

1. F. Castillo: Incremental Neural Networks: A Survey. Technical Report. INPG Grenoble. (1991)

2. F. Rosenblatt: Principles of Neurodynamics. New York:Spartan. (1962)

3. S.I. Gallant: Optimal Linear Discriminants. Proc. of the 8th. Intl. Conf. on Pattern Recognition. Vol. 2, 849-854. Paris. (1986)

4. J.A. Sirat, J.P. Nadal: Neural Trees: A New Tool for Classification. Technical Report. Laboratoires d'Electronique Philips. (1990)

5. J.P. Nadal, G. Toulouse: Information Storage in Sparsely-Coded Memory Nets. Network, Vol. 1, 67-74. (1990)

6. S. Knerr, L. Personnaz, G. Dreyfus: A new Approach to the Design of Neural Network Classifiers and its Application to the Automatic Recognition of Handwritten Digits. Proc. of IJCNN 91, 91-96, Seattle. (1991)

7. J.M. Moreno, F. Castillo, J. Cabestany: Enhanced Unit Training for Piecewise Linear Separation Incremental Algorithms. Proc. of ESANN 93, 33-38, Brussels. (1993)

8. J.M. Moreno, F. Castillo, J. Cabestany: Optimized Learning for Improving the Evolution of Piecewise Linear Separation Incremental Algorithms. New Trends in Neural Computation, 272-277, J. Mira, J. Cabestany, A. Prieto (eds.), Springer-Verlag. (1993)

9. M. Frean: The Upstart Algorithm: A Method for Constructing and Training Feedforward Neural Networks. Neural Computation 2, 198-209. (1990)