NEUROCOMPUTING : ALGORITHMS, ARCHITECTURES AND APPLICATIONS
NATO ASI Series (Springer).

SINGLE-LAYER LEARNING REVISITED : A STEPWISE PROCEDURE FOR BUILDING AND TRAINING A NEURAL NETWORK

S. Knerr, L. Personnaz, G. Dreyfus
Ecole Supérieure de Physique et de Chimie Industrielles
de la Ville de Paris
Laboratoire d'Electronique
10, rue Vauquelin
F - 75005 Paris
France

Abstract

A stepwise procedure for building and training a neural network intended to perform classification tasks, based on single layer learning rules, is presented. This procedure breaks up the classification task into subtasks of increasing complexity in order to make learning easier. The network structure is not fixed in advance: it is subject to a growth process during learning. Therefore, after training, the architecture of the network is guaranteed to be well adapted for the classification problem.

1. Introduction

We present a procedure for simultaneously building and training a neural network designed to perform classification tasks. The general motivation behind these investigations is the need for a constructive method which is flexible enough to allow an adaptation of the classifier to the distribution of the classes. Several attempts in the same spirit have been published [1,2,3,4]. Specifically, the most recent studies aim at overcoming the well-known deficiencies of the backpropagation method, namely, that the architecture (number of layers, number of neurons per layer) is fixed a priori by educated guesses. The general guideline for choosing the number of hidden units is the fact that the size of the intermediate layer(s) must be large enough to allow a proper separation of the elements of the training set, and restricted enough for the network to have a good chance to generalize properly. Moreover, the training procedure tends to be time-consuming, and the algorithm is not guaranteed to converge to a good solution, if such solutions exist. Designers usually endeavour to alleviate the task of the network by imposing constraints on the architecture and on the synaptic weights. These constraints are specific to each problem [5,6].

The stepwise construction procedure which is described in the present paper is intended to find a neural network that solves multi-class problems. It consists of several steps of increasing complexity, starting with attempts at separating linearly each class from all other classes, and subsequently separating classes pairwise with subnetworks. In the latter case, however, training involves the first layer of adaptive connections only.

In the first part, we shall present the building and training procedure; we shall subsequently describe the relevant learning rules and describe the construction of the subnetworks. Experimental results on the contiguity problem will be presented.

2. The building procedure

We assume that the problem involves c classes in R^n. The network will be made of binary state neurons. Figure 1 is a two-dimensional illustration, featuring one class (#3) which is linearly separable from all others, two classes (#1 and 4) which are pairwise linearly separable from all others, and two classes (#2 and 5) which are not linearly separable.

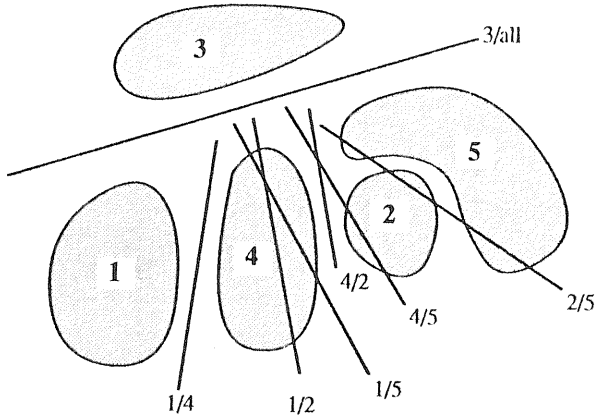


Figure 1

The **first step** of the procedure attempts to separate each class linearly from all others, using c "grandmother" neurons. After training, c_s neurons will perform 100 % correct classification of the examples, and $c_u = c\text{-}c_s$ neurons will turn out to be unsuccessful. The latter c_u neurons are discarded. In the case illustrated on Figure 1, there will be one single successful grandmother neuron.

The second step consists in attempting a pairwise separation of the c_u remaining classes, using $c_u(c_u-1)/2$ neurons, followed by a layer of AND functions with c_u-1 inputs, which can possibly be implemented in the form of c_u neurons. Training is performed on the first layer of connections only. As a result of this step, the examples corresponding to some classes will be fully correctly classified, whereas examples of other classes will still be misclassified, if the latter are not linearly separable. The AND-neurons corresponding to classes having misclassified examples are discarded, whereas all the neurons of the first layer are retained.

The third step has to deal with nonlinearly separable classes. Thus, mutilayer subnetworks must be built. At this point, two possibilities exist : either use of a multilayer Perceptron, trained by backpropagation [7], or use of a pairwise separation procedure, which will be described in a subsequent paragraph. It should be noticed that we now have to deal with the examples of the nonlinearly separable classes only. Thus, the training subset will be much smaller than the initial training set, and each subnetwork will have one output neuron only, so that training will be much easier and faster.

At this point, the structure of the network corresponding to Figure 1 is shown on Figure 2.

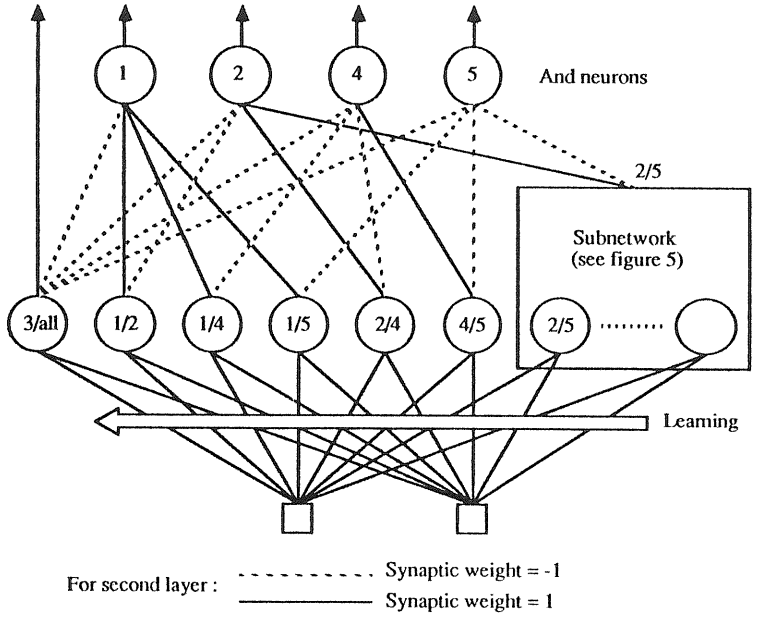


Figure 2

Before going on to discuss the procedure used to build the subnetworks, we shall briefly recall the training algorithms which can be used.

3. Single-layer training algorithms

Single-layer training algorithms have been known for a long time [8,9]. In this section, we shall recall their principles and describe their main properties. Since we consider neurons with binary states, each neuron defines a hyperplane in input space. The synaptic weights $\{C_j\}$ are computed separately for each neuron by a gradient method. In all cases, we deal with supervised learning: the training set is made of pairs $\{\underline{x}^k, d^k\}$ of example vectors $\underline{x}^k$ and their associated desired scalar outputs d^k.

Perceptron learning rule :

Neurons with binary state s = {+1,-1} are used for learning and for classification. The criterion to be minimized is

$$J(\{C_j\}) = \Sigma_M(-v^k) \quad , \quad \text{with } v^k = \Sigma_j C_j x_j^k,$$

where M is the set of misclassified examples. The bias weights are connected to a constant input unit $x_0 = +1$, and they are not shown on Figures throughout this paper.

For each example $\{\underline{x}^k, d^k\}$ presented to the network the coefficients are changed in the following way :

$$\Delta C_j = \mu\,(d^k - s^k)\,x_j^k \,.$$

with $0<\mu<1/2$. The convergence of the learning procedure is guaranteed only if the classes are linearly separable. In this case, training stops as soon as the whole training set is properly classified, so that the hyperplanes tend to lie very close to the marginal examples.

A useful variation of the perceptron learning rule is the "pocket algorithm" [10] : after presenting an example and changing the coefficients, the whole training set is tested. If the present classification results are considered to be better than any previous result, the synaptic weights are stored. At the end of the learning phase, the set of coefficients which yielded the best result is kept. This technique can accommodate a variety of criteria for the decision whether a result is better than the previous ones.

Delta rule (Widrow-Hoff) :

Neurons with binary state {+1,-1} are used. The criterion to be minimized is

$$J(\{C_j\}) = \Sigma_k(d^k - v^k)^2 \,.$$

The learning rule is expressed as :

$$\Delta C_j = \mu\,(d^k - v^k)\,x_j^k / \|\underline{x}^k\|^2 \,, \quad \text{with } 0 < \mu < 1.$$

The procedure is guaranteed to converge to the least squares solution, which does not necessarily separate the classes, even if they are linearly separable. This will happen when the numbers of examples differ widely from class to class, or if the classes have very different volumes in input space. This is especially true when grandmother cells are used, i.e. when one tries to separate one class from all other classes.

Generalized delta rule :

Neurons with sigmoidal transfer functions $s = f(v) = (\exp(v)-1)/(\exp(v)+1)$ are used for training, but binary state neurons can be used once training has been completed. The criterion to be minimized is

$$J(\{C_j\}) = \sum_k (d^k - s^k)^2 .$$

The learning rule is expressed as :

$$\Delta C_j = \mu \, (d^k - s^k) \, x_j^k \, f'(v^k)$$

In the framework of single layer training, the learning rule is guaranteed to converge to the least squares solution. If the classes are linearly separable, the criterion can be made to vanish and the hyperplane will effectively separate the examples, as opposed to the Widrow-Hoff rule. The criterion takes into account the distance of the marginal examples to the separating hyperplane, so that the position of the latter is usually more satisfactory than the position obtained by the Perceptron rule. Therefore, the generalized delta rule takes the best of both worlds, as illustrated on Figure 1, at the expense of using a sigmoidal transfer function for the neurons during learning.

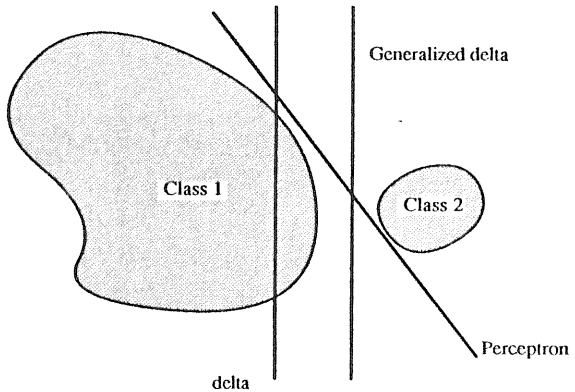


Figure 3

Independently of the learning rule used, there are some situations in which the criterion is minimum if all the examples of the training set lie on the same side of the hyperplane. In such a case, the procedure is trapped. In order to continue, one of the misclassified examples is chosen and the synaptic weights are changed in the following fashion :

$$\Delta C_j = - (1+\varepsilon) \, v^k \, x_j^k / \|\underline{x}^k\|^2 , \quad \varepsilon > 0 \text{ and small.}$$

This guarantees that example k will be properly classified, and the procedure can be resumed.

4. Subnetwork construction

We are now dealing with classes which are not pairwise linearly separable. For each such pair of classes, we shall build a subnetwork which is intended to separate them. The procedure is as follows :

- as stated above, pairwise separation was attempted at step two of the construction procedure, resulting in a neuron which failed to separate the classes (e.g. classes #2 and 5 of Figure 1); misclassified examples exist in at least one of the two regions defined by the neuron; this neuron is kept, and each region thus defined is considered separately;
- in each of these regions, we attempt to separate the misclassified examples from the correctly classified examples, thereby generating an extra neuron;
- the procedure is iterated until a satisfaction criterion is met;
- once the criterion is met, the first layer is complete; the subsequent layers perform boolean functions only, thus require no training.

The procedure is illustrated on Figure 4: starting with the neuron separating (unsuccessfully) classes 2 and 5, a neuron A is generated which tries to separate the misclassified examples of class 2 from the examples of class 5; this defines area R1, in which all examples are correctly classified, and the region which is going to be separated into areas R3 (examples correctly classified by neuron A) and R2 (examples misclassified by neuron A); therefore, a neuron B is generated, which separates the examples of area R2 from those of area R3; note that the training set for neuron B includes examples belonging to areas R2 and R3 only. A similar procedure is carried out for areas R4, R5, R6, resulting in the generation of neurons C and D.

The resulting subnetwork is illustrated on Figure 5.

Neuron labelled R3 is active if the example belongs to area R3, and is inactive otherwise; neuron R5 is active if the example belongs to area R5, inactive otherwise; hence, the output neuron of the subnetwork will be active if either R3 or R5 is active, i.e. if the input data belongs to class 2.

In this example, the procedure stops when all items of the training set are successfully classified. This may not be optimal if two classes are overlapping, in which case the procedure should be stopped before overfitting occurs.

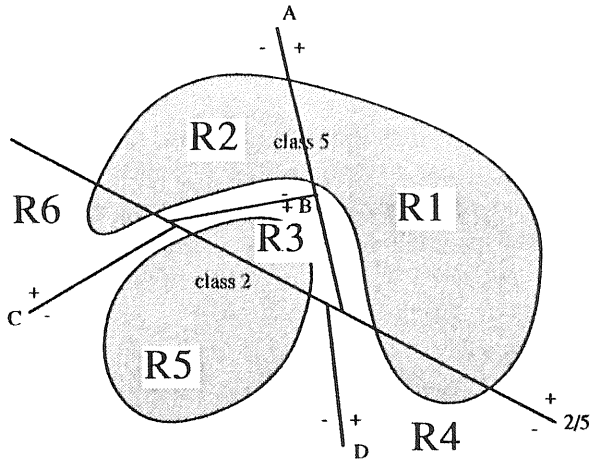


Figure 4

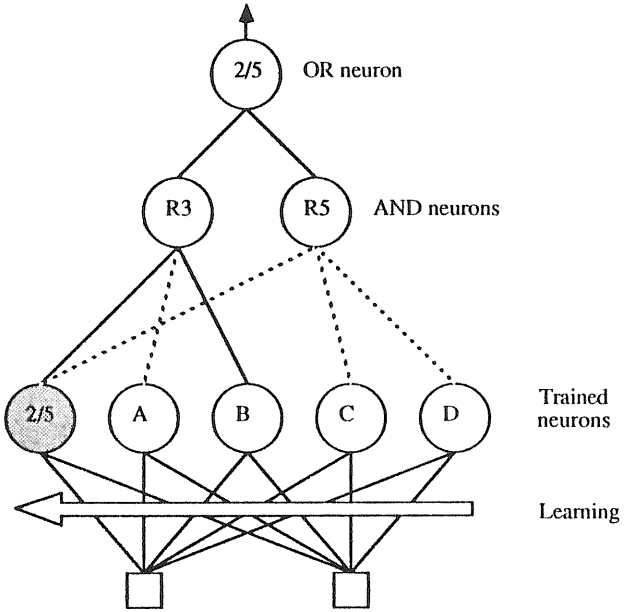


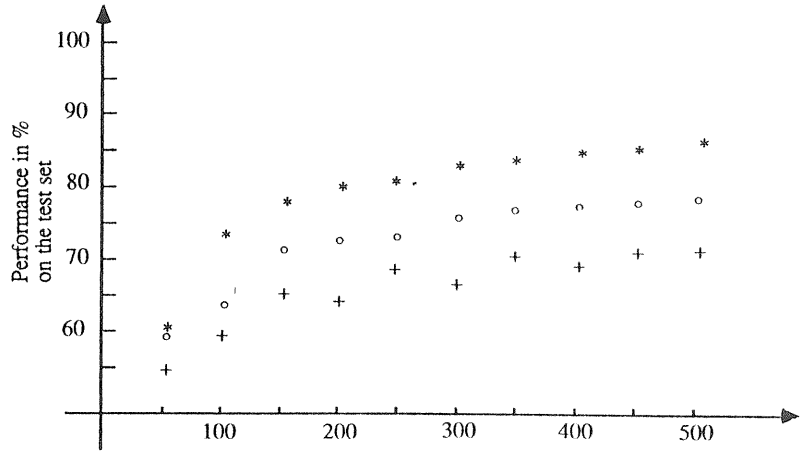
Figure 5

5. Illustrative results

As a first application of our algorithm we chose a problem with binary inputs, which is known as the "two or more clumps problem". A multilayer network trained by backpropagation finds a solution only if one gives it a hint on how the solution looks like.

The inputs to the network are one dimensional chains of binary states. This chain is in the form of a closed loop, so that the first and last pixels are neighbours. A set of contiguous black pixels is called a clump. The task is to decide whether there is only one clump in the closed chain, or if there are more. In the first case the desired output is -1; it is 1 otherwise.

Binary chains of length N=25, with a varying number of clumps, were generated. The training set and the test set had the same size, ranging from 50 to 500 examples with a given average number of clumps and the same number of examples in each class. The stopping criterion for the construction procedure was that all examples of the training set should be correctly classified. Figure 6 shows the results obtained for a given size of the training set, averaged over 10 different instances of the set.



Average number of clumps : + plot : 3 clumps
° plot : 6 clumps
* plot : 9 clumps

Figure 6

It is not surprising that results are improved when the average number of clumps becomes larger, because it is easier to discriminate a chain with 9 clumps from a chain with one clump than to discriminate a chain with 3 clumps from a chain with one clump. Obviously, the performance also improves with the size of the training set.
Our results are in good agreement with those obtained by the tiling-algorithm [3]. However, the size of the networks, and therefore the training time, compare favourably. Our networks have only one layer of adaptive connections, and two layers of explicit boolean functions, which can even be implemented by standard logic gates. For the case of 500 training examples and an average number of 3 clumps, our network has 16 neurons in the first layer only, plus 15 neurons performing a logic AND in the second layer and one neuron performing a logic OR in the third layer.
As a matter of course, the network was not expected to - and indeed did not - discover the "human" solution (the first layer of which is an edge detector). Note that the above results were obtained without any a priori knowledge of the desirable structure of the network [11].
Results concerning more typical classification problems will be published in a more detailed paper.

6. Conclusions

In the present paper it was shown that a stepwise building procedure with single-layer training offers a potentially powerful alternative to multilayer networks trained by backpropagation. Successive steps of the building procedure are of increasing complexity, thus simple tasks are solved by small networks. Since the network structure is not fixed a priori, the resulting architecture will be well adapted for the classification problem after training. Moreover, convergence of the training procedure in a relatively short time as compared to backpropagation can be guaranteed. In addition, binary state neurons can be used, which makes hardware implementations easier. The performances of the method for automatic classification of pictures and phonemes are under investigation.

References

1. Weaver C.S.: "Some Properties of Threshold Logic Unit Pattern Recognition Networks", IEEE Transactions on Computers, Vol. c-24, No. 3, 290, March 1975.
2. Rujan P., Marchand M.: "Learning by activating neurons: a new approach to learning in neural networks", KFA Jülich preprint 1988.
3. Mézard M., Nadal J.-P.: "Learning in Feedforward Networks: the Tiling Algorithm", L.P.S.E.N.S. preprint, January 1989.
4. Koutsougeras C., Papachristou C.A.: "Training of a neural Network for Pattern Classification Based on an Entropy Measure", IEEE International Conference on Neural Networks, I-247, San Diego, July 1988.
5. Waibel A., Hanazawa T., Hinton G., Shikano K., Lang K.,"Phoneme Recognition Using Time-Delay Neural Networks", IEEE Transactions on Acoustics, Speech and Signal Processing, Vol. 37, No. 3, 328, March 1989.
6. Le Cun Y.: "Constrained Networks For Handwritten Numeral Recognition", Snowbird Conference on "Neural Networks for Computing", Snowbird, April 1989.
7. Rummelhart D.E. and McCleland J.L.: "Parallel Distributed Processing", Bradford Books, Cambridge MA 1986.
8. Duda R., Hart P.: "Pattern Recognition and Scene Analysis", John Wiley & Sons 1973.
9. Minsky M., Papert S.: "Perceptrons", MIT Press, Cambridge MA 1969.
10. Gallant S.I.: "Optimal Linear Discriminants", Eighth International Conference On Pattern Recognition, Vol. 2, 849, Paris, France 1986.
11. Denker J., Schwartz D., Wittner B., Solla S., Hopfield J.J., Howard R. and Jackel L.: "Automatic Learning, Rule Extraction and Generalization", Complex Systems 1,1987.