Constructive and Robust Combination of Perceptrons

Robert Eigenmann and Josef A. Nossek
Lehrstuhl für Netzwerktheorie und Schaltungstechnik
Technische Universität München
E-mail: Robert.Eigenmann@nws.e-technik.tu-muenchen.de

Abstract

We propose a new strategy for a constructive training of feedforward neural networks to classify linearly nonseparable patterns. The algorithm results in a configuration of the first layer of the network, which is able to give a faithful internal representation of the input patterns. The weights of the network are obtained by the introduced CadaTron algorithm, which is able to separate clusters of data in a robust way. Iteratively, further neurons are added to the neural net in order to decrease the training error. Unnecessary neurons are removed, so this algorithm leads to a network with low complexity and excellent generalization properties. The results of this work are based on the classification of handwritten characters.

Keywords: *Multi Layer Perceptron, Internal Representation, Decision Boundaries, Robustness, Constructive Algorithm, Clustering*

1. Introduction

To solve a classification problem, a n dimensional input pattern has to be mapped onto a discrete value, which corresponds to the index of the different classes. By a binary coding of the class index, the K-class problem is splitted into a set of parallel 2-class problems, each giving a binary output. Neurons with a hard limiter activation function (perceptrons) are predestinated for such 2-class problems. Unfortunately, perceptrons are only able to solve linearly separable problems. To manage nonlinearly separable problems, a mulitlayer architecture is needed. Because the distribution of classes in the pattern space generally is unknown, the network architecture has to be flexible. A very complex network will not be able to generalize. On the other hand, a network with less neurons may not be able to solve the classification problem at all. Therefore, some constructive algorithms have been developed, like the *tiling algorithm* [4] or the *cascade correlation algorithm* [1]. These algorithms start with a low network complexity and insert additional neurons, if needed. All these algorithms "try to do the best they can with a few neurons" in order to decrease the training error.

In this paper a new learning algorithm is presented, which follows a completely different strategy. Instead of trying to decrease the training error, we introduce the CadaTron algorithm, which ensures a robust embedding of already correctly classified training patterns for each training epoch. It will be shown, that this leads to an efficient solution for two-class problems, if the classes consist of some nonlinearly separable clusters in the input space. The constructive algorithm inserts additional neurons or deletes them, if they turned out to be unnecessary. This algorithm is based on the geometrical properties of the defined network architecture. The efficiency of training is improved by an adequate selection of training data, based on the decision regions of the topology.

2. Properties of layered networks

To separate two disjoint classes of training pattern $\mathcal{X}^+, \mathcal{X}^- \subset \mathcal{X}$, with $\mathcal{X}^+ \cup \mathcal{X}^- = \mathcal{X}$ and $\mathcal{X}^+ \cap \mathcal{X}^- = \emptyset$, out of the finite training set $\mathcal{X} \subset X$, a mapping $\mathbf{x} \mapsto y$; $\mathbb{R}^n \rightarrow \{-1; +1\}$ has to be realized. The binary output y indicates the estimated class

$$y = \begin{cases} +1 & : \quad \mathbf{x} \in \mathcal{X}^+ \quad \text{class1} \\ -1 & : \quad \mathbf{x} \in \mathcal{X}^- \quad \text{class2} \end{cases} . \tag{1}$$

Properties of a single perceptron

Neurons with a hard-limiter activation function (perceptrons) [10, 5, 2] perform the desired mapping, given by a weighted summation of the components of the input vector $\mathbf{x}$. By keeping the n-th component of $\mathbf{x}$ constant $x_n = -1$, the threshold of the neuron is implemented with the n-th component w_n of the weight vector $\mathbf{w}$.

$$y = \operatorname{sign}\left(\mathbf{w}^T\mathbf{x}\right) = \begin{cases} +1 & ; \quad \mathbf{w}^T\mathbf{x} > 0 \\ -1 & ; \quad \mathbf{w}^T\mathbf{x} < 0 \\ 0 & ; \quad \text{else} \end{cases} . \tag{2}$$

1015-4651/96 $5.00 © 1996 IEEE
Proceedings of ICPR '96

Equation (2) defines a $n-1$ dimensional hyperplane E in $\mathbb{R}^n$. The output y indicates, on which side of E the input $\mathbf{x}$ is situated. Let $\mathcal{R}^+$ ($\mathcal{R}^-$) be a subset of $\mathcal{X}$, containing the relevant training patterns out of class 1 (class 2) which **should be**, [1] and $\mathcal{P}^+$ ($\mathcal{P}^-$) be the patterns, which **actually are** on the positive (negative) side of E,

$$\mathcal{P}^+ = \{\mathbf{x} \in \mathcal{R} | \mathbf{w}^T\mathbf{x} > 0\}; \quad \mathcal{P}^- = \{\mathbf{x} \in \mathcal{R} | \mathbf{w}^T\mathbf{x} < 0\}, \tag{3}$$

whereas the disjoint subsets $\mathcal{G}$ and $\mathcal{B}$ are called the sets of 'good' and 'bad' patterns.

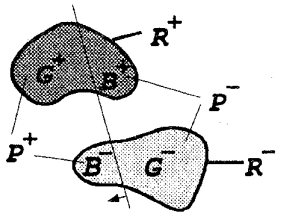


Figure 1. The subsets of 'good' and 'bad' patterns with respect to the perceptron problem and its solution

$$\begin{array}{rclrcl} \mathcal{R}^+ \cap \mathcal{P}^+ &=& \mathcal{G}^+ & \mathcal{R}^- \cap \mathcal{P}^- &=& \mathcal{G}^- \\ \mathcal{R}^+ \cap \mathcal{P}^- &=& \mathcal{B}^+ & \mathcal{R}^- \cap \mathcal{P}^+ &=& \mathcal{B}^- \\ \mathcal{G}^+ \cup \mathcal{G}^- &=& \mathcal{G} & \mathcal{B}^+ \cup \mathcal{B}^- &=& \mathcal{B} \\ \mathcal{G} \cap \mathcal{B} &=& \emptyset & \mathcal{G} \cup \mathcal{B} &=& \mathcal{R} = \mathcal{P} \end{array}$$

A layer of perceptrons

A layer of h neurons performs a nonlinear mapping $\mathbb{R}^n \rightarrow \{-1,+1\}^h$ by a weight matrix $\mathbf{W} = (\mathbf{w}_1, \mathbf{w}_2, \ldots \mathbf{w}_h)$, consisting of the weight vectors $\mathbf{w}_i$ of each neuron. Because of the h defined hyperplanes, the input space is splitted into 2^h regions (if $h \leq n$), which we call **cells**. The cell number $k = 0 \ldots 2^h - 1$ corresponds to the output vector $\mathbf{z}$

$$\mathbf{z} = \operatorname{sign}\left(\mathbf{W}^T\mathbf{x}\right) \in \{-1,+1\}^h. \tag{4}$$

of the perceptron layer, if $\mathbf{z}$ is interpreted as a binary number with the component z_1 as the MSB. For example, the 6-th cell corresponds to the output $\mathbf{z} = (+1,+1,-1)^T \rightsquigarrow \text{bin}(\mathbf{z}) = 110_{\text{bin}} = 6$.

Let $\mathcal{C}_k$ be the subset of patterns within cell k and let $\mathcal{C}_k^+$ ($\mathcal{C}_k^-$) be the patterns within cell k and belonging to class1 (class2)

$$\mathcal{C}_k = \{\mathbf{x} \in \mathcal{X} | \quad \text{bin}(\mathbf{z}) = k\}, \tag{5}$$

$$\mathcal{C}_k^+ = \mathcal{C}_k \cap \mathcal{X}^+, \tag{6}$$

$$\mathcal{C}_k^- = \mathcal{C}_k \cap \mathcal{X}^-. \tag{7}$$

Therefore, a value c_k is assigned to each cell i, indicating the estimated class of the patterns in this cell. In [11] a statistically based rule to determine c_k out of $\mathcal{C}_k^+$ and $\mathcal{C}_k^-$ is given, i.e. for the case of balanced classes $|\mathcal{X}^+| = |\mathcal{X}^-|$

$$c_k = \begin{cases} +1 & ; \quad |\mathcal{C}_k^+| > |\mathcal{C}_k^-| \\ -1 & ; \quad |\mathcal{C}_k^+| < |\mathcal{C}_k^-| \\ 0 & ; \quad \text{else} \end{cases}. \tag{8}$$

The value $c_k = 0$ is taken, if no pattern is within cell k ("don't care") or if $|\mathcal{C}_k^+| = |\mathcal{C}_k^-|$ ("don't know"). In this case the cell k is not able to make a statement about the class. Figure 2(b) shows a possible partial linear separation of X, defined by the 3 hyperplanes and the values c_k for $k = 0 \ldots 7$.

According to (3), the set $\mathcal{P}_i^+$ ($\mathcal{P}_i^-$) containes the pattern, situated on the positive (negative) side of the hyperplane E_i.

$$\mathcal{P}_i^+ = \{\mathbf{x} \in \mathcal{R}_i | \mathbf{w}_i{}^T\mathbf{x} > 0\}; \quad \mathcal{P}_i^- = \{\mathbf{x} \in \mathcal{R}_i | \mathbf{w}_i{}^T\mathbf{x} < 0\}. \tag{9}$$

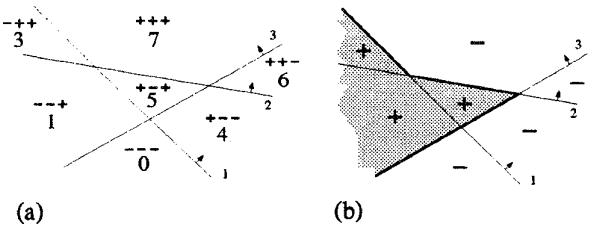


Figure 2. h hyperplanes in the pattern space X. (a) shows the enumeration of the cells and (b) shows an example, how to separate the pattern space by nonlinear decision boundaries (here the number of cells is less than 2^h, because $h = 3 > n = 2$).

2.1. Relevant training patterns for a neuron

A line segment $\subset E_i$ separating two cells $(k1, k0)$ is called an edge, whereas the binary values of $k1$ and $k0$ differ in the i-th bit. An edge is element of the decision boundary, if the values c_{k1} and c_{k0} differ in the sign. Those edges are printed with bold lines in Figure 2(b). A couple of cells $(k1, k0)$ are **relevant cells for neuron** i, if the edge separating these cells is part of the decision boundary and a line segment of the hyperplane E_i.

$$(k1,k0): \begin{cases} k1 &=& (b_1 \ldots b_{i-1}\ 1\ b_{i+1} \ldots b_h)_{\text{bin}} \\ k0 &=& (b_1 \ldots b_{i-1}\ 0\ b_{i+1} \ldots b_h)_{\text{bin}} \\ c_{k1} \cdot c_{k0} &=& -1 \end{cases}$$

$$\forall b_j \in \{0;1\} \tag{10}$$

[1] For a single neuron, $\mathcal{R}^+ = \mathcal{X}^+$ and $\mathcal{R}^- = \mathcal{X}^-$, but we may take $\mathcal{R} \subset \mathcal{X}$ for better learning properties.

The set of relevant training patterns for the i-th neuron

$$\mathcal{R}_i = \bigcup_{(k1,k0) \text{ out of eqn. (10)}} (C_{k1} \cup C_{k0}) \qquad (11)$$

is splitted into $\mathcal{R}_i^+$ and $\mathcal{R}_i^-$, indicating the side of the hyperplane, on which the patterns should be embedded. Thereby, 'good' patterns are left of the actual side of E_i and 'bad' patterns are to be embedded on the other side of the hyperplane. With the subsets $\mathcal{G}_k$ ($\mathcal{B}_k$) for the 'good' ('bad') patterns within cell k

$$\mathcal{G}_k = \begin{cases} C_k \cap \mathcal{X}^+ & ; \quad c_k = +1 \\ C_k \cap \mathcal{X}^- & ; \quad c_k = -1 \end{cases} \qquad (12)$$

$$\mathcal{B}_k = C_k \setminus \mathcal{G}_k, \qquad (13)$$

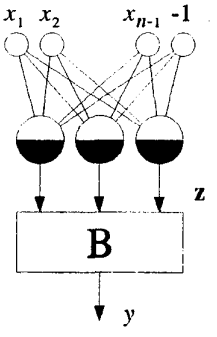
and the sets $\mathcal{G}$ and $\mathcal{B}$ over all cells are given by

$$\mathcal{G} = \bigcup_{k=0}^{2^h-1} \mathcal{G}_k \qquad \mathcal{B} = \bigcup_{k=0}^{2^h-1} \mathcal{B}_k, \qquad (14)$$

the relevant training pattern for the neuron i are

$$\begin{aligned} \mathcal{R}_i^+ &= \mathcal{R}_i \cap \left((\mathcal{P}_i^+ \cap \mathcal{G}) \cup (\mathcal{P}_i^- \cap \mathcal{B})\right) \\ \mathcal{R}_i^- &= \mathcal{R}_i \cap \left((\mathcal{P}_i^- \cap \mathcal{G}) \cup (\mathcal{P}_i^+ \cap \mathcal{B})\right). \end{aligned} \qquad (15)$$

The internal representation $\mathbf{z}$ is called to be 'faithful' [4], if each cell containes only patterns of a unique class (or no pattern at all). The remaining training error is given by $\varepsilon = \frac{|\mathcal{B}|}{|\mathcal{X}|}$.



$$\mathbf{z} = \text{sign}\left(\mathbf{W}^T\mathbf{x}\right)$$

$$\mathbf{B} : \mathbf{z} \mapsto y$$

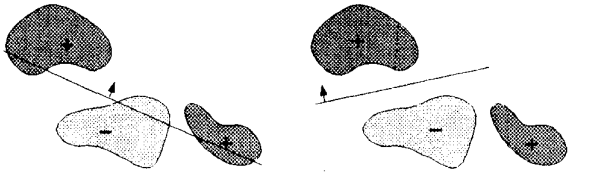
Figure 3. Network architecture

Figure 3 shows the complete network architecture. $\mathbf{B}$ is a Boolean function defined by the values c_k and maps the internal representation $\mathbf{z}$ onto a binary value y.

3. Training of a single neuron

We introduce a new weight adaptation rule for perceptrons, called the CadaTron (Clustering AdaTron) algorithm. Usually, perceptrons are trained by Hebbian learning rules [7, 12] in order to minimize the training error. But these algorithms are failing, if the training set is not linearly separable. For such problems, a criterion for stopping the algorithm has to be defined.

In Figure 4 a motivating example of a linearly nonseparable database is given. A single perceptron minimizing the training error is shown in (a). Although the problem could be solved with 2 neurons, solution (a) makes it impossible to find a second hyperplane that solves the problem. The CadaTron algorithm will find a hyperplane shown in (b). Although the training error is higher than in (a), a second hyperplane could easily lead to a separation of the training patterns.



(a) (b)

Figure 4. (a): weight found by minimizing the training error. (b): weight found by the CadaTron algorithm

Consider a 2-class problem for one neuron, where $\mathcal{R}^+$ has to be separated from $\mathcal{R}^-$. The initial weight vector $\mathbf{w}(0)$ may be obtained by two randomly choosen patterns $\mathbf{x}^+ \in \mathcal{R}^+$ and $\mathbf{x}^- \in \mathcal{R}^-$

$$\mathbf{w}(0) = \mathbf{x}^+ - \mathbf{x}^-. \qquad (16)$$

According to this weight vector, the 'good' patterns $\mathcal{G}(0)$ are robustified, utilizing the AdaTron algorithm [7], whereby the AdaTron solves the optimization problem

$$\max_{\mathbf{w}} \min_{i=1\ldots P} \mathbf{w}^T\xi_i \quad \text{subject to} \quad ||\mathbf{w}|| = 1, \qquad (17)$$

for P given training patterns ξ_i, with

$$\xi_i = \begin{cases} \mathbf{x}_i & \text{for} \quad \mathbf{x}_i \in \mathcal{R}^+ \\ -\mathbf{x}_i & \text{for} \quad \mathbf{x}_i \in \mathcal{R}^- \end{cases}.$$

If the training set is linearly separable, the AdaTron will find a robust and unique solution for $\mathbf{w}$. Taking $\mathcal{G}^+$ and $\mathcal{G}^-$ to train the neuron, linear separability of the training set is guaranteed. The weight vector $\mathbf{w}(1)$, found by the AdaTron separates $\mathcal{G}^+(0)$ and $\mathcal{G}^-(0)$ in a robust way, and there may be even some additional patterns $\Delta\mathcal{G}(0) \in \mathcal{B}(0)$, which are supplementarily classified correctly after robustification of the 'good' patterns. Iteratively, with the iteration index k, the new set of 'good' patterns is $\mathcal{G}(k+1) = \mathcal{G}(k) \cup \Delta\mathcal{G}(k)$ with $\Delta\mathcal{G}(k) \subset \mathcal{B}(k)$. If $\Delta\mathcal{G}(k) \neq \emptyset$, training continues with $\mathcal{G}(k+1)$, otherwise the algorithm terminates.

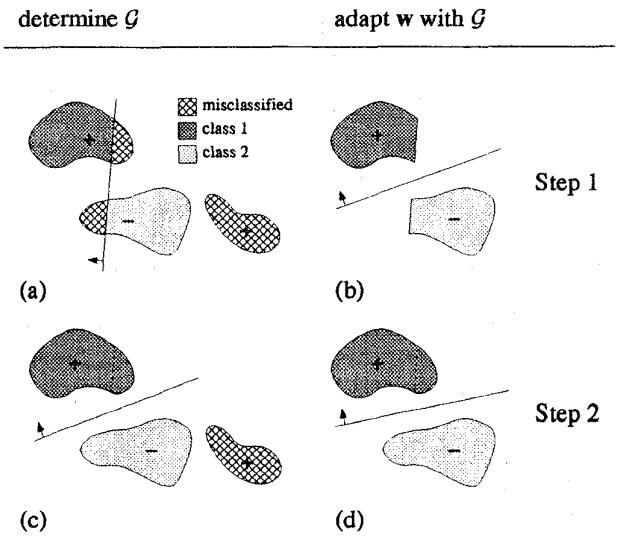


Figure 5. An example for the CadaTron learning

w = CadaTron($\mathcal{R}^+, \mathcal{R}^-, \mathbf{w}(0)$)
determine $\mathcal{G}^+(0), \mathcal{G}^-(0)$
$k = 0$
do
  $\mathbf{w}(k+1)$ = AdaTron($\mathcal{G}^+(k), \mathcal{G}^-(k), \mathbf{w}(k)$)
  determine $\mathcal{G}^+(k+1), \mathcal{G}^-(k+1)$
  $k = k + 1$
while ($|\mathcal{G}(k)| > |\mathcal{G}(k-1)|$)

The convergence of this algorithm is ensured, because the convergence of the AdaTron algorithm is guaranteed by the choice of the training patterns and the algorithm stops, if $\Delta\mathcal{G} = \emptyset$. Figure 5 shows the behavior of the CadaTron algorithm. In (a) the patterns $\mathcal{G}(0)$ are determined and utilized to train the neuron (b). Then with the new weight $\mathbf{w}(1)$, the patterns $\mathcal{G}(1)$ are determined (c). The AdaTron finds the weight vector $\mathbf{w}(2)$ shown in (d) with those training patterns. Then, no "additional good patterns" are found, so the iteration stops.

4. Constructive training of the network

In this section we discuss, how the CadaTron algorithm is embedded in a constructive modification of the network architecture. The property of the CadaTron, not to care about the misclassified patterns, requires a growing of the perceptron layer, in order to decrease the training error. The constructive algorithm is called *CCA* (Constructive Clustering AdaTron).

Training the hidden layer

Consider a neural network with already h neurons in the hidden layer, shown in Figure 3. The easiest way to train the neurons is to adapt the weights of one neuron after the other. The set of training patterns $\mathcal{R}_i$ for the neuron i has to be determined out of (11), each time before training this neuron. In a loop for $i = 1 \ldots h$, the neurons are trained with the CadaTron to separate the subsets $\mathcal{R}_i^+$ and $\mathcal{R}_i^-$, determined by (15).

Inserting a neuron

If $\mathbf{w}_i(k) = \mathbf{w}_i(k-1) \forall_{i=1\ldots h}$, and $\mathcal{B} \neq \emptyset$, an additional neuron is inserted. This neuron is trained to separate the patterns $\mathcal{G}_m$ from $\mathcal{B}_m$ in the cell m, that containes the most misclassified patterns.

$$\begin{aligned} |\mathcal{B}_m| \geq |\mathcal{B}_k| \qquad & \forall_{k \neq m;\ k=0\ldots 2^h-1} \\ \mathcal{R}_{h+1}^+(0) = \mathcal{C}_m^+ \qquad & \mathcal{R}_{h+1}^-(0) = \mathcal{C}_m^- \end{aligned} \tag{18}$$

The initial weight $\mathbf{w}_{h+1}(0)$ is obtained out of (16). After the training of $\mathcal{R}_{h+1}^+(0)$; $\mathcal{R}_{h+1}^-(0)$, the Boolean mapping **B** has to be redefined by determining the values c_k for the 2^{h+1} cells.

Deleting a neuron

If a neuron i turns out to have no relevant patterns $\mathcal{R}_i = \emptyset$, it can be omitted without any influence on the training error. The values c_k for the 2^{h-1} cells have to be determined again after the removal of the neuron.

Constructive training: CCA algorithm

DO
  DO
    FOR $i = 1 \ldots h$
      determine $\mathcal{R}_i^+, \mathcal{R}_i^-$ eqn.(11)
      IF $\mathcal{R}_i^+ = \emptyset \vee \mathcal{R}_i^- = \emptyset$
      THEN
        delete neuron i
        $h = h - 1$
        recalculate **B** eqn.(8)
      ELSE
        $\mathbf{w}_i(k+1)$ = CadaTron($\mathcal{R}_i^+, \mathcal{R}_i^-, \mathbf{w}_i(k)$)
    NEXT i
    $k = k + 1$
  WHILE $\mathbf{w}_i(k) \neq \mathbf{w}_i(k-1) \forall_{i=1\ldots h}$
  IF $\mathcal{B} \neq \emptyset$
  THEN
    $h = h + 1$
    insert neuron h
    find cell m with the maximal error
    determine $\mathcal{R}_h^+, \mathcal{R}_h^-$ eqn.(18)
    initialize $\mathbf{w}_h(0)$ eqn.(16)
    $\mathbf{w}_h(1)$ = CadaTron($\mathcal{R}_h^+, \mathcal{R}_h^-, \mathbf{w}_h(0)$)
    recalculate **B** eqn.(8)
WHILE $\mathcal{B} \neq \emptyset$

5. Results

In this section, the results obtained with the presented constructive algorithm are compared with the results of the *MadaTron* (Multilayer AdaTron) algorithm [6, 9]. The MadaTron algorithm calculates the weights of a neural network with committee machine architecture, shown in Figure 3 with **B** equal to the majority function $\mathbf{B}_{\text{maj}} : y = \text{sign}(\sum z_i)$. This algorithm requires a fixed size for the hidden layer and is known to give excellent results for the application of classification compared to other well-known algorithms [9]. The number of hidden layers is set to be $h = 5$. In order to keep the complexity of both networks comparable, the CCA is prohibited to insert more than 5 neurons.

A training set of 10×1000 greylevel bitmaps of handwritten characters $0 \ldots 9$ is taken out of the database of NIST [8]. The *Hough*-transformation [3] is applied to obtain 40 features out of the bitmap. A coding of the class index with the 7-bit Hamming code [6] gives 7 parallel 2-class problems for the bits $b = 0 \ldots 6$, which are trained separately. Each bit has been trained 20 times to get the mean training error $\bar{\varepsilon}$ and the standard deviation σ, shown in Table 1.

	MadaTron		CCA	
	$\bar{\varepsilon}$ [%]	σ [$\times 10^{-3}$]	$\bar{\varepsilon}$ [%]	σ [$\times 10^{-3}$]
bit 0	1.61	1.3	3.58	3.2
bit 1	3.09	2.7	4.24	7.3
bit 2	4.74	3.7	4.62	8.0
bit 3	7.90	5.3	6.62	5.7
bit 4	6.32	6.2	5.47	7.0
bit 5	4.94	3.3	5.26	8.5
bit 6	6.09	8.3	5.68	9.7
time[2]	$\approx$ 40 min.		$\approx$ 7 min.	

Table 1. Training error

CCA improves in 4 of 7 bits compared to MadaTron. The error of the other bits is within an acceptable range. Comparing the time for training, an improvement by a factor of $5 \ldots 6$ is achieved with CCA.

The classification error on 10×500 test pattern, which were not included in the training set, was 10.43% for the MadaTron and 9.2% for the CCA on zero rejection.

6. Conclusion

In this paper we presented a new constructive algorithm to obtain the first layer of a feedforward neural network and the Boolean mapping of this layer to a binary output. This network architecture is applied to 2-class problems.

The CadaTron (Clustering AdaTron) algorithm has been introduced. With this algorithm, one single neuron can be trained in order to separate two clusters in a robust manner. The convergence of this algorithm is guaranteed by taking a linear subset of the training patterns to be learned with the AdaTron algorithm.

With the CCA (Constructive Clustering AdaTron) algorithm, each neuron of the hidden layer is trained iteratively. The training patterns are splitted into several linearly separable subsets by reorganizing the decision regions with respect to the geometrical properties of the topology. Neurons are inserted or deleted, so the algorithm finds an accurate size for the network to solve the classification problem.

The introduced algorithm has been compared with the MadaTron, which is known to be an excellent algorithm to solve classification problems.

[2] Training of a single bit on a SPARC10 workstation

References

[1] S. E. Fahlman and C. Lebiere. The cascade correlation learning architecture. Technical Report CMU-CS-90-100, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213, Aug. 1991.

[2] G. J. Gibson and C. F. N. Cowan. On the decision regions of multilayer perceptrons. *Proc. of the IEEE*, 78(10):1590–1594, oct 1990.

[3] R. C. Gonzalez and P. Wintz. *Digital Image Processing*. Addison Wesley, Reading, MA, 1987.

[4] M. Mézard and J.-P. Nadal. Learning in feedforward layered networks: the tiling algorithm. *J. Phys. A: Math. Gen.*, 22:2191–2203, 1989.

[5] M. L. Minksy and S. Papert. *Perceptrons*. M.I.T.-Press, Cambridge, expanded edition, 1988.

[6] P. Nachbar. *Entwurf robuster neuronaler Netze*. PhD thesis, Technical University Munich, Institute for Network Theory and Circuit Design, 1994.

[7] P. Nachbar, J. Strobl, and J. Nossek. The generalized adatron algorithm. In *International Symposium on Circuits and Systems*, volume 4, pages 2152–2156. IEEE, 1993.

[8] The state-of-the-art in OCR is subject of NIST/Census conference. *Intelligence*, 9(6):1–5, 1992.

[9] J. Nossek, P. Nachbar, and A. Schuler. Comparison of Algorithms for Feedforward Multilayer Neural Nets. ISCAS, Atlanta, to be published, 1996.

[10] F. Rosenblatt. *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanism*. Sparta, Washington, D.C., 1961.

[11] W. Utschick and J. Nossek. Bayesian Adaptation of Hidden Layers in Boolean Feedforward Neural Networks. 1996. 13th International Conference on Pattern Recognition, Vienna.

[12] B. Widrow and M. Lehr. 30 Years of Adaptive Neural Networks: Perceptron, Madaline, and Backpropagation. *Proc. of the IEEE*, 78(9):1415–1441, 1990.