Convergence Rate in a Destructive Neural Network With /Without Thresholds in The Output Layer

Yasmeen T. Mahmoud
Faculty of Science
Department of Computer Science
Assiut University
yasmenaa_t@yahoo.com

Abstract

Neural networks are a practicable solution for the extraction of accurate knowledge, where the data being mined can be so noisy, due to their relatively good tolerance to noisy and generalization ability and the performance of a neural network is directly related to its parameters and architecture.
The degree of complexity of ANN increases exponentially as a factor of the numbers of input and hidden nodes. The complexity problem can be improved by constructing the structure of the network based on constructive learning and destructive learning.
So, the objective of the network is to learn or to discover some involvement between input and output patterns to find the structure of the input patterns. The learning process is achieved through the modification of the connection weights between units. It is known as the network's topology which determines the network's final behavior.
The goal of finding an optimal topology is to minimize an error function while conserving generalization capabilities.
In our work, the destructive topology is used: first the algorithm trains a big size of neural network on the data, and then prunes it to increase its generalization capability while preserving its accuracy.
The present paper introduces destructive neural network learning techniques and presents the analysis of the convergence rate of the error in a neural network with and without threshold in the output layer.
The present paper introduces destructive neural network learning techniques and presents the analysis of the convergence rate of the error in a neural network with and without threshold.

Keywords
Neural Network, Destructive Learning, Pruning.

1. INTRODUCTION

Artificial Neural Network (ANN) is a powerful data modeling tool for classification and learning method [1,2,3,4].
Mainly, neural networks are built from simple units, sometimes called neurons or cells by analogy with the real thing. These units are connected by a set of weighted connections. Each unit codes or corresponds to an attribute or a characteristic of a pattern that we want to analyze or that we want to use as a predictor. Learning is usually skillful by modification of the connection weights.
These networks usually organize their units into several layers. The first layer is called the input layer, the last one the output layer. The intermediate layers (if any) are called the hidden layers. The information to be analyzed is supplied to the neurons of the first layer and then propagated to the neurons of the second layer for additional processing. The result of this processing is then propagated to the next layer and so on until the last layer. Each unit receives some information from other units and processes this information, which will be transformed into the output of the unit.
Usually, the neural network connectivity and the activation functions are fixed by the human designer and the only parameters for model selection are the number of neurons for each layer and, in some occasions, the number of layers. Those parameters are then optimized with respect to the dataset.
During learning phase, the topology of the neural network plays a significant role in whether or not the network can be trained to learn a particular dataset [5,6,7].
A simple topology will result in a network that cannot learn to approximate a complex function, while a large topology is probable to result in a network losing its generalization capability. This loss of generalization is the result of overfitting the training data: instead of approximating a function present in the data, a neural network that has an excessively complex structure may have the ability to memorize the dataset, allowing noise within the data to be learned, resulting in imprecise predictions on future samples.
In this work, we study the situation of neural network after learning by destructive learning, and the analysis of the convergence rate of the error in a neural network with and without threshold.

2. Destructive Learning

Network pruning presents a projecting approach for dynamically determining an appropriate network topology. Pruning techniques [8,9,10] begin by training a larger than necessary network and then reduce weights and neurons that are deemed redundant. Constructive algorithms present several significant advantages over pruning based algorithms including the ease of specification of the initial network topology, better economy in terms of training time and number of training examples, and potential for converging to a smaller network with superior generalization [11,12,13].

3. Convergence Rate in a Destructive Neural Network without Thresholds in the Output Layer

In this section we will present the analysis of the convergence of the error in destructive neural network.
Let $\boldsymbol{v}$ denote a connection weight vector between the $(h+1)$-th hidden unit and the input layer, and let w be a weight vector connecting the $(h + 1)$-th hidden unit with the output layer , The ANN after removing the $(h + 1)$-th hidden unit is shown in Fig.1 We denote the weight matrices $(V, \boldsymbol{v})$ and (W, w) by $\tilde{V}$and $\widetilde{W}$, respectively.
Then we can write the output error between the outputs of the network for the inputs x^v and the relevant outputs y^v
as

$$J(V,W) = \sum_{v=1}^{m} \left\| g^{-1}(y^v) - \widetilde{W} f(Vx^v) - \boldsymbol{w} f(\boldsymbol{v}.x^v) \right\|^2$$

where

$$Wf(Vx^v) = \widetilde{W} f(Vx^v) + wf(\boldsymbol{v}.x^v)$$

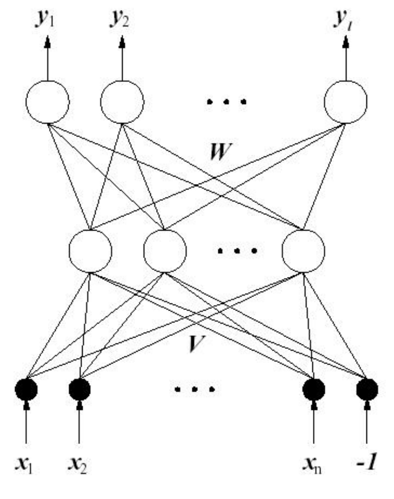

Fig.1: Neural network without threshold and after remove one unit from the hidden layer.

and w is the removing weight vector.The function can be written as follows:

$$J = J(\tilde{V}, \widetilde{W}) - 2\langle d, w\rangle + a\|w\|^2 \qquad (1)$$

in which $d$and $a$denote

$$d = \sum_{v=1}^{m} f(v.x^v)c^v$$

and

$$a = \sum_{v=1}^{m} f^2(v.x^v),$$

where $c^v = g^{-1}(y^v) - Wf(Vx^v)$ and the symbol $\langle .,.\rangle$ indicates an inner product inR^l. When the vector v is fixed, the vector w which minimizes the error function by

$$w = \frac{d}{a}$$

So the error after removing a hidden unit can be expressed as

$$\begin{aligned}\tilde{c}^v &= g^{-1}(y^v) - \widetilde{W}f(\tilde{V}x^v)\\ &= g^{-1}(y^v) - Wf(Vx^v) + wf(vx^v)\\ &= c^v + \frac{d}{a}f(v.x^v).\end{aligned}$$

Furthermore, using the symbols $\tilde{C}_i^t = (c_i^1, c_i^2, \dots, c_i^m)$, $C_i^t = (c_i^1, c_i^2, \dots, c_i^m)$ and $S^t = (S_1, S_2, \dots, S_m)$ with $s_v = f(v.x^v)$, we have

$$\tilde{C}_i^t = C_i^t + \frac{1}{a}C_i^t SS^t\,. \qquad (2)$$

Moreover we can rewrite (2) as

$$\tilde{C}_i^t = C_i^t\left(I_m + \frac{1}{a}SS^t\right) = C_i^t \Gamma_1(v),$$

where I_m is the unit matrix and

$$\Gamma_1(v) = I_m + \frac{SS^t}{a}.$$

It is easy to show that the matrix $\Gamma_1(v)$satisfies

$$1 \le \frac{\langle \Gamma_1(v)U, U\rangle}{\|U\|^2} \le 2$$

for arbitrary vector U as follows:

This shows that the error after determining the weight w between the hidden and output layers is not convergent. In addition, the eigenvalues of this matrix can be obtained as follows:

One propriety of the matrix $\Gamma_1(v)$ is

$$\langle \Gamma_1(v)U,U\rangle = \|U\|^2 + \frac{1}{a}\langle S,U\rangle^2$$
$$\leq \|U\|^2 + \frac{1}{a}\|S\|^2\|U\|^2 = 2 \qquad (3)$$

since $a = \|S\|^2$, and the other is

$$\langle \Gamma_1(v)U,U\rangle = \|U\|^2 + \frac{1}{a}\langle S,U\rangle^2$$
$$\geq \|U\|^2 \qquad (4)$$

From (3) and (4) we get

$$1 \leq \frac{\langle \Gamma_1(v)U,U\rangle}{\|U\|^2} \leq 2$$

which implies

$$1 \leq max\,\lambda_v \leq 2$$

Where $\lambda_v,\ v = 1,\dots,m$ are the eigenvalues of the matrix $\Gamma_1(v)$.
This shows that the error after determining the weight **w** between the hidden and output layers is not convergent. In addition, the eigenvalues of this matrix can be obtained in details to examine the Convergence rate of the error in more detail. The matrix $\Gamma_1(v)$can be written as

$$\Gamma_1(v) = \begin{pmatrix} z_{11} & -z_{12} & \cdots & -z_{1m} \\ -z_{21} & z_{22} & \cdots & -z_{2m} \\ \cdots & \cdots & \cdots & \cdots \\ -z_{m1} & -z_{m2} & \cdots & z_{mm} \end{pmatrix},$$

Where
$z_{ii} = 1 + s_i^2/a,\ z_{ij} = -s_i s_j/a$, and $z_{ij} = z_{ji}$.
The characteristic equation of this matrix is

$$\gamma_1(\lambda) = \begin{vmatrix} \lambda - z_{11} & z_{12} & \cdots & z_{1m} \\ z_{21} & \lambda - z_{22} & \cdots & z_{2m} \\ \cdots & \cdots & \cdots & \cdots \\ z_{m1} & z_{m2} & \cdots & \lambda - z_{mm} \end{vmatrix} = 0.$$

By putting $\eta_i = (1-\lambda)\,a/s_i^2$, we can reform $\gamma_1(\lambda)$as follows:

$$\gamma_1(\lambda) = \frac{-\prod_{i=1}^{m} s_i^2}{a^m} L_1(\lambda) = 0,$$

where

$$L_1(\lambda) = \begin{vmatrix} \eta_1 + 1 & 1 & \dots & 1 \\ 1 & \eta_2 + 1 & \dots & 1 \\ \cdots & \cdots & \cdots & \cdots \\ 1 & 1 & \dots & \eta_m + 1 \end{vmatrix}.$$

The determinant $L_1(\lambda)$can be transformed by Laplace theorem as

$$L_1(\lambda) = \prod_{v=1}^{m} \eta_v + \sum_{i=1}^{m} \prod_{v=1}^{i-1} \eta_v \prod_{v=i+1}^{m} \eta_v$$
$$= \prod_{v=1}^{m} \eta_v + \sum_{i=1}^{m} \frac{\prod_{v=1}^{m} \eta_v}{\eta_i}.$$

Thus, we have

$$\gamma_1(\lambda) = \frac{-\prod_{i=1}^{m} s_i^2}{a^m}\left(\prod_{v=1}^{m} \eta_v + \sum_{i=1}^{m} \frac{\prod_{v=1}^{m} \eta_v}{\eta_i}\right)$$
$$= (2-\lambda)(1-\lambda)^{m-1} = 0.$$

Hence the eigenvalues of this matrix are given by

$$\lambda = 2, 1, \dots, 1, 1.$$

This shows that the error after determining the weight w between the hidden and output layers is not convergent.

4. Convergence Rate in a Destructive Neural Network With Thresholds in the Output Layer

We consider the network with thresholds $\theta_i, i = 1, 2, \dots, l,$ in its output layer before removing one unit in the hidden layer as shown in Fig.2 In this case, we can write

$$y_i = g\left(\sum_{j=1}^{h} w_{ij}\, f\left(\sum_{k=1}^{n+1} v_{jk} x_k\right) - \theta_i\right), i = 1, 2, \dots, l$$

whose simple form is

$$y = g(Wf(Vx) - \theta),$$

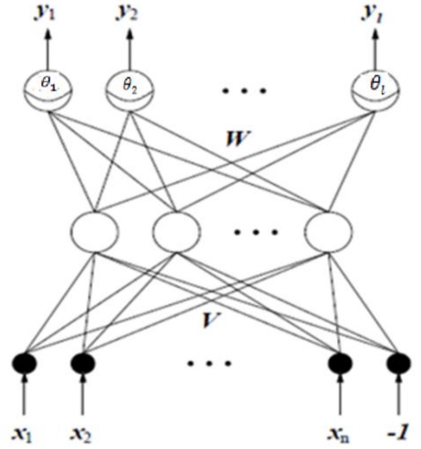

Fig.2: Neural network with threshold and before removing one unit in the hidden layer.

Where *$\Theta = (\Theta_1, \Theta_2, \dots, \Theta_l)$* and *$g(Wf(Vx)-\Theta) = (g(W_1.f(Vx)-\Theta_1), (g(W_2.f(Vx)-\Theta_2), \dots, (g(W_i.f(Vx)-\Theta_l)$*. The error function that related to the present network takes the following form

$$J(V, W, \theta) = \sum_{v=1}^{m} \|g^{-1}(y^v) - Wf(Vx^v) + \theta\|^2.$$

We remove one unit from the hidden layer and represent removed weight vectors again by $v$and w. By removing one hidden unit, we can write the threshold vector θ *as* $\theta = \tilde{\theta} + \Delta\theta$. The network after removing one hidden unit is shown in Fig.3. The error function related to this network can be expressed as

$$J(\tilde{V}, \widetilde{W}, \tilde{\theta}) = \sum_{v=1}^{m} \left\|g^{-1}(y^v) - \widetilde{W}f(\tilde{V}x^v) + \tilde{\theta}\right\|^2,$$

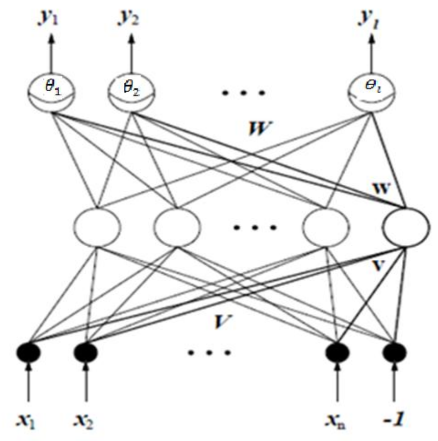

Fig.3: Neural network with threshold and after removing one unit in the hidden layer.

where we have used again the symbols $\tilde{V} = (V, v)$ and $\widetilde{W} = (W, w)$. The same procedure as in the previous section will be applied in order to determine w and $\Delta\theta$ so that $J(\tilde{V}, \widetilde{W}, \tilde{\theta})$ is minimum.
Since $Wf(Vx^v) = \widetilde{W}f(Vx^v) + wf(v.x^v)$ and $\theta = \tilde{\theta} + \Delta\theta$,we can decompose the error function $J(\tilde{V}, \widetilde{W}, \tilde{\theta})$ as

$$J(\tilde{V}, \widetilde{W}, \tilde{\theta}) = J(V, W, \theta) + 2\sum_{v=1}^{m} f(v.x^v)\langle q^v + \Delta\theta, w\rangle$$

$$-\sum_{\nu=1}^{m} f^2(v.x^\nu)\|w\|^2 - 2\sum_{\nu=1}^{m}\langle\Delta\theta, q^\nu\rangle - \sum_{\nu=1}^{m}\|\Delta\theta\|^2,$$

where

$$q^\nu = g^{-1}(y^\nu) - Wf(Vx^\nu) + \theta.$$

By minimizing this error function, we can easily determine w and $\Delta\theta$ as

$$w = \frac{md_1 - a_2 d_2}{b}, \tag{5}$$

$$\Delta\theta = \frac{a_2 d_1 - a_1 d_2}{b}, \tag{6}$$

where

$$a_1 = \sum_{\nu=1}^{m} f^2(v.x^\nu),$$

$$a_2 = \sum_{\nu=1}^{m} f(v.x^\nu),$$

$$d_1 = \sum_{\nu=1}^{m} f(v.x^\nu)\, q^\nu,$$

$$d_2 = \sum_{\nu=1}^{m} q^\nu,$$

$$b = ma_1 - a_2^2.$$

Now we consider the convergence rate of the error as in the previous section. Since

$$\begin{aligned}\tilde{q}^\nu &= g^{-1}(y^\nu) - \widetilde{W}f\left(\tilde{V}x^\nu\right) + \tilde{\theta} \\ &= g^{-1}(y^\nu) - Wf(Vx^\nu) + wf(v.x^\nu) + \theta - \Delta\theta \\ &= q^\nu + wf(v.x^\nu) - \Delta\theta,\end{aligned}$$

we can write the error at *i*-th component in the output layer as

$$\tilde{q}_i^\nu = q_i^\nu + w_i f(v.x^\nu) - \Delta\theta_i. \tag{7}$$

From (5), the *i*-th component of *w* can be written as

$$\begin{aligned}w_i &= \frac{m}{b}d_{1i} - \frac{a_2}{b}d_{2i} \\ &= \frac{m}{b}\sum_{\nu=1}^{m} s_\nu q_i^\nu - \frac{a_2}{b}\sum_{\nu=1}^{m} q_i^\nu \\ &= \frac{m}{b}\ {}^tQ_i S - \frac{a_2}{b}\ {}^tQ_i 1 \\ &= \ {}^tQ_i\left[\frac{m}{b}S - \frac{a_2}{b}1\right].\end{aligned}$$

Similarly from (6), we can also write $\Delta\theta_i$as

$$\Delta\theta_i = \ {}^tQ_i \left[\frac{a_2}{b} S - \frac{a_2}{b} 1\right],$$

Where

${}^t\tilde{Q}_i = (\tilde{q}_i^1, \tilde{q}_i^2, \dots, \tilde{q}_i^m),\ {}^tQ_i = (q_i^1,\ q_i^2, \dots, q_i^m),\ {}^tS = (s_1,\ s_2, \dots, s_m)$, and ${}^t1 = (1, 1, \dots, 1)$ with $s_v = f(v.x^v)$.

Hence, (7) can be rewritten as

$$\tilde{q}_i^v = q_i^v + \frac{m}{b}\ {}^tQ_i S s_v - \frac{a_2}{b}\ {}^tQ_i 1 s_v - \frac{a_2}{b}\ {}^tQ_i S + \frac{a_1}{b}\ {}^tQ_i 1$$

or,

$${}^t\tilde{Q}_i = \ {}^tQ_i \left[I_m + \frac{m}{b} S\ {}^tS - \frac{a_2}{b} S 1\ {}^tS - \frac{a_2}{b}\ {}^t1 + \frac{a_1}{b} 1\ {}^t1\right] = \ {}^tQ_i \Gamma_2(v),$$

where

$$\Gamma_2(v) = I_m + \frac{m}{b} S\ {}^tS - \frac{a_2}{b} S 1\ {}^tS - \frac{a_2}{b}\ {}^t1 + \frac{a_1}{b} 1\ {}^t1.$$

We will show that the matrix $\Gamma_2(v)$satisfies the stability condition

$$1 \le \frac{\langle \Gamma_2(v) U, U \rangle}{\|U\|^2} \le 2$$

for arbitrary vector U.

One propriety of the matrix $\Gamma_2(v)$ is

$$\begin{aligned}\langle \Gamma_2(v) U, U \rangle &= \|U\|^2 + \frac{m}{b}\langle S, U\rangle^2 - \frac{2a_2}{b}\langle S, U\rangle\langle 1, U\rangle + \frac{a_1}{b}\langle 1, U\rangle^2 \\ &\le \|U\|^2 + \frac{m}{b}\|S\|^2\|U\|^2 \\ &\le 2\|U\|^2. \qquad (8)\end{aligned}$$

and the other is

$$\begin{aligned}\langle \Gamma_2(v) U, U \rangle &\ge \|U\|^2 + \frac{m}{b}\left(\langle S, U\rangle - \frac{a_2}{m}\langle 1, U\rangle\right)^2 + \frac{1}{m}\|1\|^2\|U\|^2 \\ &= \frac{m}{b}\left(\langle S, U\rangle - \frac{a_2}{m}\langle 1, U\rangle\right)^2 + \frac{m}{b} \ge \frac{m}{b}\left\|S - \frac{a_2}{m} 1\right\|^2 \|U\|^2 \\ &= \|U\|^2. \qquad (9)\end{aligned}$$

From (8) and (9) we get

$$1 \le max\, \lambda_v \le 2,$$

Where $\lambda_v, v = 1, \dots, m$ are the eigenvalues of the matrix $\Gamma_2(v)$.

Next, we calculate the eigenvalues of the matrix $\Gamma_2(v)$to examine the Convergence rate of the error in more detail. The matrix $\Gamma_2(v)$can be written as

$$\Gamma_2(v) = \begin{pmatrix} z_{11} & z_{12} & \cdots & z_{1m} \\ z_{21} & z_{22} & \cdots & z_{2m} \\ \cdots & \cdots & \cdots & \cdots \\ z_{m1} & z_{m2} & \cdots & z_{mm} \end{pmatrix},$$

Where

$z_{ii} = 1 + \frac{ms_i^2}{b} - \frac{2a_2 s_i}{b} + \frac{a_1}{b}$, $z_{ij} = \frac{ms_i s_j}{b} - \frac{a_2(s_i+s_j)}{b} + \frac{a_1}{b}$, and $z_{ij} = z_{ji}$.

The characteristic equation of this matrix is

$$\gamma_2(\lambda) = \begin{vmatrix} \lambda - z_{11} & -z_{12} & \cdots & -z_{12} \\ -z_{21} & \lambda - z_{11} & \cdots & -z_{2m} \\ \cdots & \cdots & \cdots & \cdots \\ -z_{m1} & -z_{m2} & \cdots & \lambda - z_{mm} \end{vmatrix}$$

$$= \begin{vmatrix} \lambda - 1 + p_{11} & p_{12} & \cdots & p_{1m} \\ p_{21} & \lambda - 1 + p_{22} & \cdots & p_{2m} \\ \cdots & \cdots & \cdots & \cdots \\ p_{m1} & p_{m2} & \cdots & \lambda - 1 + p_{mm} \end{vmatrix}$$

$$= \begin{vmatrix} \lambda - 1 + p_{11} & p_{12} & \cdots & p_{1m} & 1 \\ p_{21} & \lambda - 1 + p_{22} & \cdots & p_{2m} & 1 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ p_{m1} & p_{m2} & \cdots & \lambda - 1 + p_{mm} & 1 \\ 0 & 0 & \cdots & 0 & 1 \end{vmatrix} = 0,$$

where

$p_{ii} = -mu_i^2/b - 1/m$, and $p_{ij} = -\,mu_i u_j/b - 1/m$, with $u_i = s_i - a_2/m$.

By putting $t_v = b(1-\lambda)/(mu_v^2) - 1$, we can reform $\gamma_2(\lambda)$ as follows:

$$\gamma_2(\lambda) = \frac{\prod_{i=1}^m u_i^2}{m}\left(\frac{m}{b}\right)^{m-1} L_2(\lambda)$$

with

$$L_2(\lambda) = \begin{vmatrix} t_1 & 1 & 1 & \cdots & 1 & 1 & \frac{1}{u_1} \\ 1 & t_2 & 1 & \cdots & 1 & 1 & \frac{1}{u_2} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 1 & 1 & 1 & \cdots & t_{m-1} & 1 & \frac{1}{u_{m-1}} \end{vmatrix}$$

$$\begin{vmatrix} 1 & 1 & 1 & \dots & 1 & t_m & \frac{1}{u_m} \\ \frac{1}{u_1} & \frac{1}{u_2} & \frac{1}{u_3} & \dots & \frac{1}{u_{m-1}} & \frac{1}{u_m} & \frac{-m^2}{b} \end{vmatrix}$$

Using the Laplace theorem, we get

$$\gamma_2(\lambda) = \frac{\prod_{i=1}^{m} u_i^2}{m}\left(\frac{m}{b}\right)^{m-1}\left[-\frac{m^2}{b}A + B\right], \qquad (10)$$

where

$$A = \begin{vmatrix} t_1 & 1 & \dots & 1 \\ 1 & t_2 & \dots & 1 \\ \dots & \dots & \dots & \dots \\ 1 & 1 & \dots & t_m \end{vmatrix}$$

and

$$B = -\sum_{i=1}^{m}\frac{1}{u_i^2}T_i + \sum_{i=1}^{m}\sum_{\substack{j=1 \\ j\neq i}}^{m}\frac{1}{u_i u_j}T_{ij}$$

With

$$T_i = \begin{vmatrix} t_1 & 1 & \dots & 0 & \dots & 1 \\ 1 & t_2 & \dots & 0 & \dots & 1 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ & & \dots & & & \\ 0 & & \dots & 1 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ & & \dots & & & \\ 1 & 1 & \dots & 0 & \dots & t_m \end{vmatrix} \quad and \quad T_{ij} = \begin{vmatrix} t_1 & 1 & \dots & 0 & \dots & 1 & \dots & 1 \\ 1 & t_2 & \dots & 0 & \dots & 1 & \dots & 1 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 1 & \dots & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 1 & 1 & \dots & 0 & \dots & 1 & \dots & 1 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 1 & 1 & \dots & 0 & \dots & 1 & \dots & t_m \end{vmatrix}$$

Generally, we have

$$\begin{vmatrix} t_1 & a & \dots & a & a \\ a & t_2 & \dots & a & a \\ \dots & \dots & \dots & \dots & \dots \\ a & a & \dots & t_{m-1} & a \\ a & a & & a & t_m \end{vmatrix} = \begin{vmatrix} t_{1-a} & 0 & \dots & 0 & a \\ a - t_2 & t_2 - a & \dots & 0 & a \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & t_{m-1} - a & a \\ 0 & 0 & \dots & a - t_m & t_m \end{vmatrix}$$

$$= \prod_{v=1}^{m}(t_v - a) + a\sum_{i=1}^{m}\prod_{v=1}^{i-1}(t_v - a)\prod_{v=i+1}^{m}(t_v - a).$$

According to the above result we can reform A,T_i, and T_i as follows:

$$A = \prod_{v=1}^{m}(t_v - 1) + \sum_{i=1}^{m}\prod_{v=1}^{i-1}(t_v - 1)\prod_{v=i+1}^{m}(t_v - 1),$$

$$T_i = \frac{\prod_{v=1}^{m}(t_v - 1)}{(t_i - 1)} + \sum_{\substack{j=1\\ j\neq i}}^{m}\frac{\prod_{v=1}^{m}(t_v - 1)}{(t_j - 1)(t_i - 1)},$$

and

$$T_{ij} = \frac{\prod_{v=1}^{m}(t_v - 1)}{(t_i - 1)(t_j - 1)}.$$

But we have $t_v = b(1-\lambda)/(mu_v^2) - 1$ and so it follows that

$$A = (2-\lambda)(1-\lambda)^{m-1}\left(\frac{b}{m}\right)^m\prod_{v=1}^{m}\frac{1}{u_v^2} \qquad (11)$$

and

$$B = -\sum_{i=1}^{m}\frac{1}{u_i^2}\frac{\prod_{v=1}^{m}(t_v - 1)}{(t_i - 1)} - \sum_{i=1}^{m}\frac{1}{u_i^2}\sum_{\substack{j=1\\ j\neq i}}^{m}\frac{\prod_{v=1}^{m}(t_v - 1)}{(t_j - 1)(t_i - 1)}$$

$$+\sum_{i=1}^{m}\sum_{\substack{j=1\\ j\neq 1}}^{m}\frac{1}{u_iu_j}\frac{\prod_{v=1}^{m}(t_v - 1)}{(t_j - 1)(t_i - 1)}$$

$$= \prod_{v=1}^{m}\frac{1}{u_v^2}(\frac{b}{m}(1-\lambda))^{m-2}\left[-m(1-\lambda)\frac{b}{m} - \sum_{i=1}^{m}\sum_{\substack{j=1\\ j\neq i}}^{m}u_j^2 + \sum_{i=1}^{m}\sum_{\substack{j=1\\ j\neq 1}}^{m}u_iu_j\right]$$

$$= K\left[-(1-\lambda)b - b + \frac{b}{m} - \frac{b}{m}\right]$$

$$= -bK(2-\lambda), \qquad (12)$$

where

$$K = \left(\frac{b}{m}(1-\lambda)\right)^{m-2}\prod_{v=1}^{m}\frac{1}{u_v^2}.$$

By substituting (11) and (12) into (10), we have

$$\gamma_2(\lambda) = -K(2-\lambda)^2 \left(\frac{m}{b}\right)^{m-2} \frac{1}{m} \prod_{i=1}^{m} u_i^2 = 0$$

which implies

$$(2-\lambda)^2(1-\lambda)^{m-2} = 0,$$

and finally we obtain

$$\lambda = 2, 2, 1, 1 \ldots, 1.$$

This theorem shows that the error after determining the weight **w** and the correction $\Delta\theta$ between the hidden and output layers is not convergent.

5. Conclusion

An **Artificial Neural Network** (ANN) is an adaptive system that changes its structure based on external or internal information that flows through the network during the learning phase. It also defined as a computational device that consists of many simple connected units (neurons) that work in parallel. The connections between the units or nodes are usually weighted by real-valued weights. Weights are the primary means of learning in neural networks, and a learning algorithm is usually used to adjust the weights.

The objective of the network is to learn or to discover some involvement between input and output patterns, to analyze, or to find the structure of the input patterns. The learning process is achieved through the modification of the connection weights between units.This construction of nodes and connections, known as the network’s topology, together with the weights of the connections, determines the network’s final behavior.

Given a neural network topology and a training set, it is possible to optimize the values of the weights in order to minimize an error function by means of any backpropagation–based algorithm or standard optimization techniques.

Producing or deciding what will be a suitable network topology is a task that is usually solved with heuristic algorithms or left to human experts. The process of “manually” design such a topology is iterative, often requires a certain amount of expertise, and it is definitively tiresome.

The problem of finding an optimal topology can be thought of as a search problem, where the search space is the space of all possible network topologies, and where the goal is to minimize an error function while conserving generalization capabilities.

More often a *destructive* search is used: first the algorithm trains a big size of neural network on the data, and then prunes it to increase its generalization capability while preserving its accuracy.

The main issue with this kind of approach is the choice of the initial network that has to be big "enough" in order to be effectively pruned.
So in this paper, destructive learning techniques have been given and a theoretical analysis of the convergence rate of the error of an artificial neural network with and without threshold that is learned by destructive learning has been also presented.

6. References

1. J. Han, M. Kamber, Data Mining: Concepts and Techniques, second ed., Mo rgan Kaufmann, 2006.
2. G. Bologna, A model for single and multiple knowledge-based networks, Ar tificial Intelligence in Medicine 2 (28) (2003) 141–163.
3. U. Markowska-Kaczmar, M. Chumieja, Discovering the mysteries of neural networks, International Journal of Hybrid Intelligent Systems 3,4 (1) (2004) 153–163.
4. W. Craven, Extracting Comprehensible Models from Trained Neural Networ ks, Ph.D. Thesis, Department of Computer Sciences, University of Wisconsi n-Madison, 1996.
5. K.J. Cios, G. William, Uniqueness of medical data mining, Artificial Intelli-gence in Medicine 1 (26) (2002) 1–24.
6. L. Ma, K. Khorasani, A new strategy for adaptively constructing multilayer f eedforward neural networks, Neurocomputing (51) (2003) 361–385.
7. R. Krishnan, G. Sivakumar, P. Bhattacharya, A search technique for rule ext raction from trained neural networks, Pattern Recognition Letters 3 (20) (19 99) 273–280.
8. H. Tsukimoto, Extracting rules from trained neural networks, IEEE Transac-tions on Neural Networks 2 (11) (2000) 377–389.
9. T.-Y. Kwok, D.-Y. Yeung, Constructive algorithms for structure learning in feedforward neural networks for regression problems, IEEE Transactions on Neural Networks 3 (8) (1997) 630–645.
10. T.-Y. Kwok, D.-Y. Yeung, Objective functions for training new hidden units in constructive neural networks, IEEE Transactions on Neural Networks 5 (8) (1997) 1131–1148.
11. L. Ma, K. Khorasani, Input-Side training in constructive neural networks bas ed on error scaling and pruning, in: Proceedings of the IJCNN, vol. VI, 2000 , pp. 455–460.

12. L. Ma, K. Khorasani, New training strategies for constructive neural networ ks with application to regression problems, Neural Networks 4 (17) (2004) 5 89–609.
13. M.A. Potter, A genetic cascade-correlation learning algorithm, in: Interna- ti onal Workshop on Combinations of Genetic Algorithms and Neural Net- wo rks, IEEE Computer Society Press, 1992, pp. 123–133.