



UNIVERSITÄT
KOBLENZ · LANDAU

Institut für Wirtschafts-
und Verwaltungsinformatik



FB 4
Informatik

Comparing the efficiency of serial and parallel algorithms for training artificial neural networks using computer clusters

Oleg V. Kryuchin
Alexander A. Arzamastsev
Klaus G. Troitzsch

Nr. 13/2011

**Arbeitsberichte aus dem
Fachbereich Informatik**

Die Arbeitsberichte aus dem Fachbereich Informatik dienen der Darstellung vorläufiger Ergebnisse, die in der Regel noch für spätere Veröffentlichungen überarbeitet werden. Die Autoren sind deshalb für kritische Hinweise dankbar. Alle Rechte vorbehalten, insbesondere die der Übersetzung, des Nachdruckes, des Vortrags, der Entnahme von Abbildungen und Tabellen – auch bei nur auszugsweiser Verwertung.

The “Arbeitsberichte aus dem Fachbereich Informatik“ comprise preliminary results which will usually be revised for subsequent publication. Critical comments are appreciated by the authors. All rights reserved. No part of this report may be reproduced by any means or translated.

Arbeitsberichte des Fachbereichs Informatik

ISSN (Print): 1864-0346

ISSN (Online): 1864-0850

Herausgeber / Edited by:

Der Dekan:

Prof. Dr. Grimm

Die Professoren des Fachbereichs:

Prof. Dr. Bátori, Prof. Dr. Burkhardt, Prof. Dr. Diller, Prof. Dr. Ebert,
Prof. Dr. Furbach, Prof. Dr. Grimm, Prof. Dr. Hampe, Prof. Dr. Harbusch,
jProf. Dr. Kilian, Prof. Dr. von Korflesch, Prof. Dr. Lämmel, Prof. Dr. Lautenbach,
Prof. Dr. Müller, Prof. Dr. Oppermann, Prof. Dr. Paulus, Prof. Dr. Priese,
Prof. Dr. Rosendahl, Prof. Dr. Viorica Sofronie-Stokkermans, jProf. Dr. Scherp Prof.
Dr. Schubert, Prof. Dr. Staab, Prof. Dr. Steigner, Prof. Dr. Sure, Prof. Dr. Troitzsch,
Prof. Dr. Walsh, Prof. Dr. Wimmer, Prof. Dr. Zöbel

Kontakt Daten der Verfasser

Oleg V. Kryuchin, Alexander A. Arzamastsev, Klaus G. Troitzsch,
Institut für Wirtschafts- und Verwaltungsinformatik

Fachbereich Informatik

Universität Koblenz-Landau

Universitätsstraße 1

D-56070 Koblenz

Email : kryuchov@gmail.com, arz_sci@mail.ru, kgt@uni-koblenz.de

Comparing the efficiency of serial and parallel algorithms for training artificial neural networks using computer clusters

Oleg V. Kryuchin, Alexander A. Arzamastsev, Klaus G. Troitzsch

Abstract

An estimation of the number of multiplication and addition operations for training artificial neural networks by means of consecutive and parallel algorithms on a computer cluster is carried out. The evaluation of the efficiency of these algorithms is developed. The multilayer perceptron, the Volterra network and the cascade-correlation network are used as structures of artificial neural networks. Different methods of non-linear programming such as gradient and non-gradient methods are used for the calculation of the weight coefficients.

Keywords: parallel algorithms, artificial neural networks, estimation of algorithm efficiency, computer clusters.

1 Introduction

1.1 Background and aims

Nowadays artificial neural networks (ANNs) are used in different branches of science and engineering, such as systems of artificial intelligence, image identification tasks, mathematical simulation (in situations when it is necessary to develop a model of an object when the structure of this object cannot be easily defined by simple rules of this science branch), computer vision systems, analysis and forecast of time series, simulation of social objects etc. [1, 2, 3].

Thus an ANN is a universal instrument for building mathematical models for objects which are defined by empirical data. The main limitations of ANNs lie in the compilation of algorithms for structure selection and in the large size of the vector of variables of the target function which is to be minimized by the algorithm training the ANN. These limitations define that the development of an ANN-model which is sufficient to model a certain object may take a large amount of time.

One of the most advanced methods of solving this problem is the development of parallel algorithms of building an ANN-model using computer clusters [4, 5, 6]. The traditional techniques of estimating the speed of algorithms count the multiplication and addition operations executed by these algorithms. So the aim of this paper is the calculation of the numbers of multiplication and addition operations for serial and parallel algorithms training the ANN. These numbers may be used for examining parallel algorithms .

1.2 Used network structure types

This subsection will consider a few types of ANNs. These are the multilayer perceptron (MLP) shown in Figure 1, a cascade-correlation network shown in Figures 2–5) and a Volterra network shown in Figure 6. These structure types were selected because they are very often used for the approximation and the forecasting of time series [3, 7].

A multilayer perceptron (MLP) is a feedforward ANN model that maps sets of input data into a set of appropriate output. An MLP consists of multiple layers of nodes in a directed graph which is fully connected from one layer to the next. Except for the input nodes, each node is a neuron (or processing element) with a nonlinear activation function. Usually an MLP utilizes a supervised learning technique called backpropagation for training the network but it may also use other algorithms [8, 9]. The MLP is a modification of the standard linear perceptron, which can distinguish data which are not linearly separable [10].

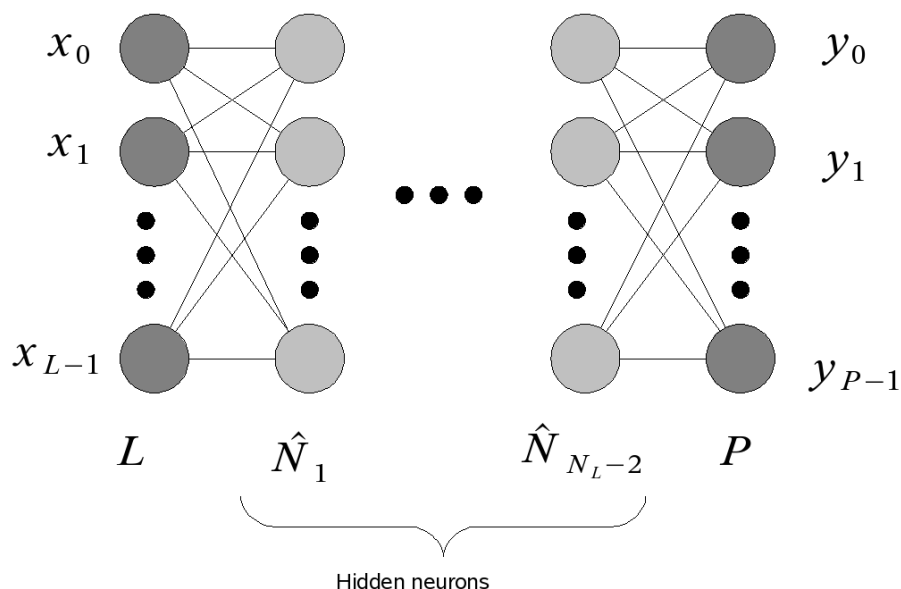


Figure 1: The multilayer perceptron.

In Figure 1 \vec{x} is the vector of input data and \vec{y} is the output data vector. This network has N_L layers. The first layer is the input layer and consists of L neurons. The last layer is the output layer and consists of P neurons. The other layers are hidden and the i -th hidden layer consists of \hat{N}_i neurons.

A cascade-correlation network is a particularized multilayer neural network construction which was recommended by Fahlman. In this network the selection of the structure and the search for weight coefficients are executed in parallel. In each training step one new hidden neuron is added. So training such network always consists of the selection of the structure and of updating weight coefficients (and sometimes searching neuron activation functions). The architecture of the cascade-correlation network is the consolidation of neurons by links, and the view of this consolidation is the progressive cascade. Each added neuron connects to output nodes and to all hidden neurons. All input neurons and hidden neurons connect to each output neuron, too [11, 12].

The training of this network begins before hidden neurons are added. Weight coefficients (and activation functions) are searched for minimizing the target function value (Fig. 2).

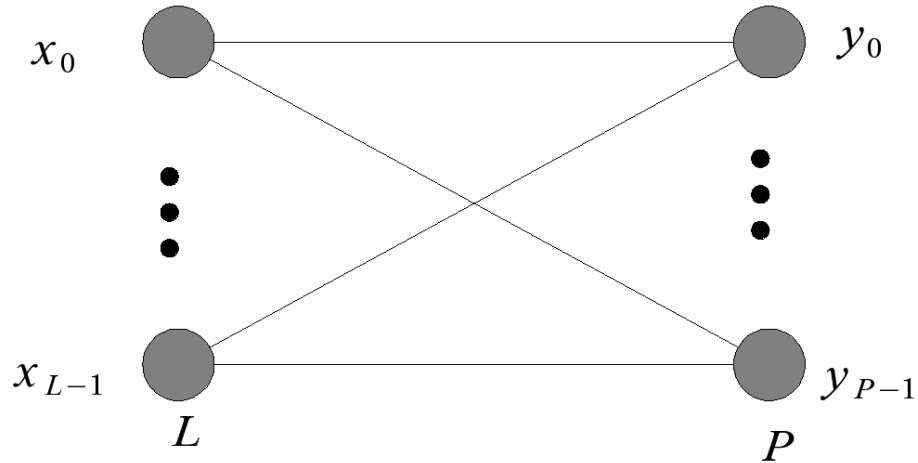


Figure 2: The cascade-correlation network without hidden neurons.

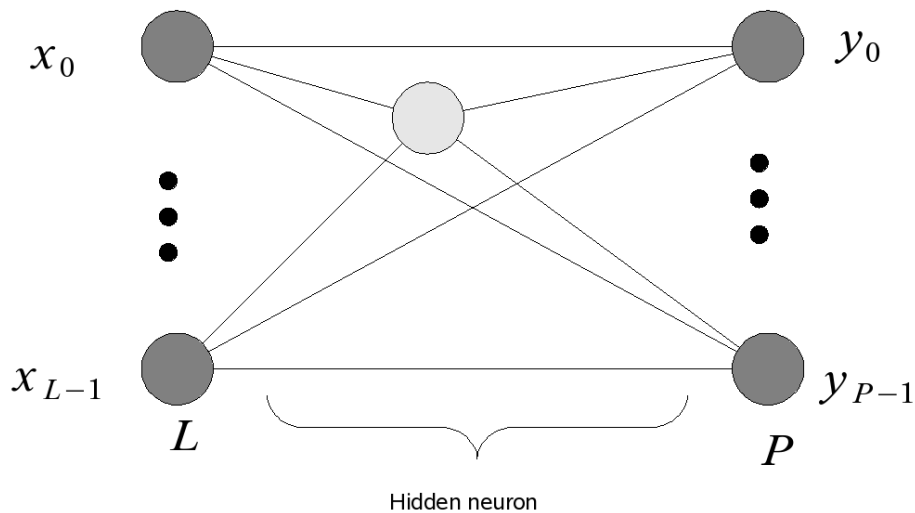


Figure 3: The cascade-correlation network with one hidden neuron.

If after training a result is considered reasonable with respect to the feasible or desired accuracy then the process of the training and the network structure formed finishes. Otherwise it is necessary to add a new hidden neuron (Fig. 3). For this a special procedure is used. At first input weights of the new neuron are formed, initialized, updated and fixed. Then the output weights of this neuron are connected to all output neurons. After this step new weight coefficients are searched. If the result of the training

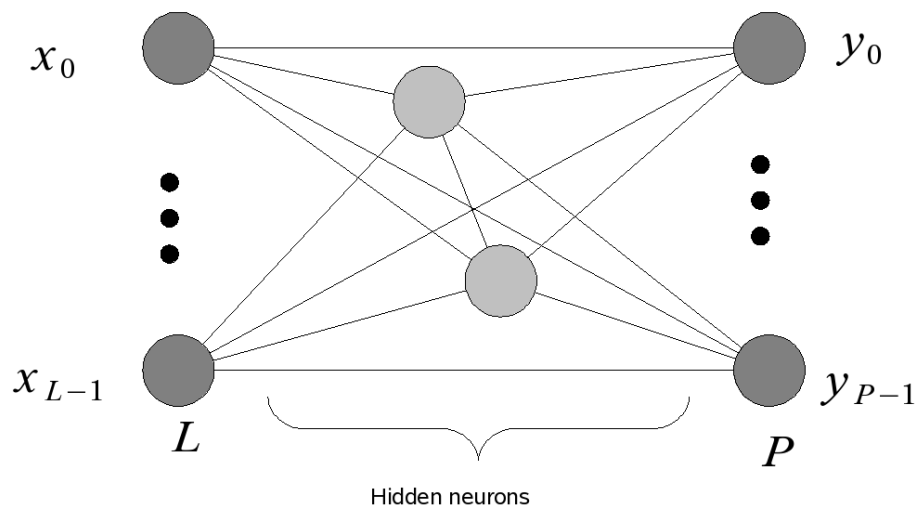


Figure 4: The cascade-correlation network with two hidden neurons.

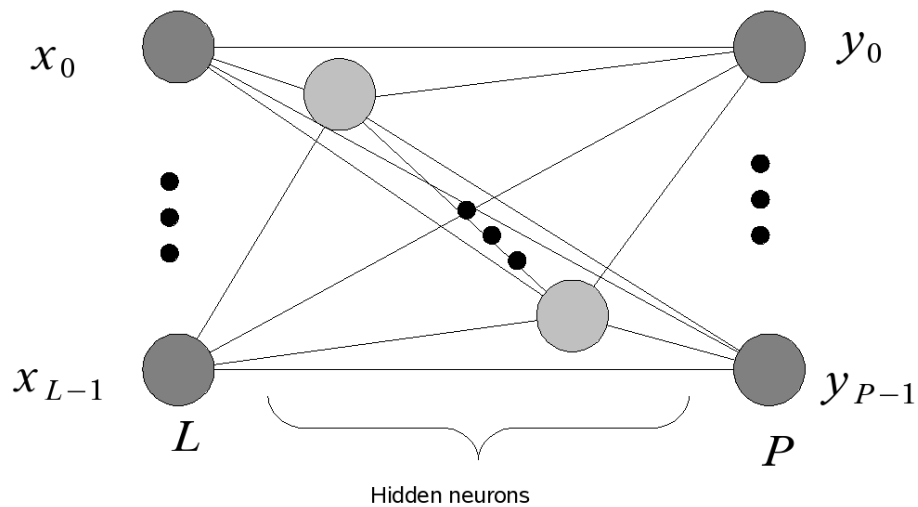


Figure 5: The cascade-correlation network with few hidden neurons.

is reasonable then the training is finished, otherwise again a new neuron is added. This process continues until the desired result of the training is obtained (pic. 4-5) [8, 14].

A Volterra network is a dynamic network for the nonlinear adaptation of an array of signals following each other. The input vector $\vec{x}(t)$ (which is shown in formula (1)) activates the network at moment i . As the Volterra series is defined, the output signal

$y(i)$ is calculated by formula (2).

$$\vec{x}(t) = (x_t, x_{t-1}, \dots, x_{t-L}) \quad (1)$$

$$y(t) = \sum_{i_1=1}^L (w_{i_1} x_{t-i_1}) + \sum_{i_1=1}^L \left(\sum_{i_2=1}^L (w_{i_1, i_2} x_{t-i_1} x_{t-i_2}) \right) + \dots + \quad (2)$$

$$+ \sum_{i_1=1}^L \left(\sum_{i_2=1}^L \left(\sum_{i_3=1}^L \dots \sum_{i_K=1}^L (w_{i_1, i_2, \dots, i_K} x_{t-i_1} x_{t-i_2} \dots x_{t-i_K}) \right) \right)$$

where weights w_{i_1} , w_{i_1, i_2} , w_{i_1, i_2, i_3} , w_{i_1, i_2, i_3, i_4} etc. are called Volterra kernels and correspond to the reaction of the highest factors. This polynomial degree is called the Volterra series degree [3, 15].

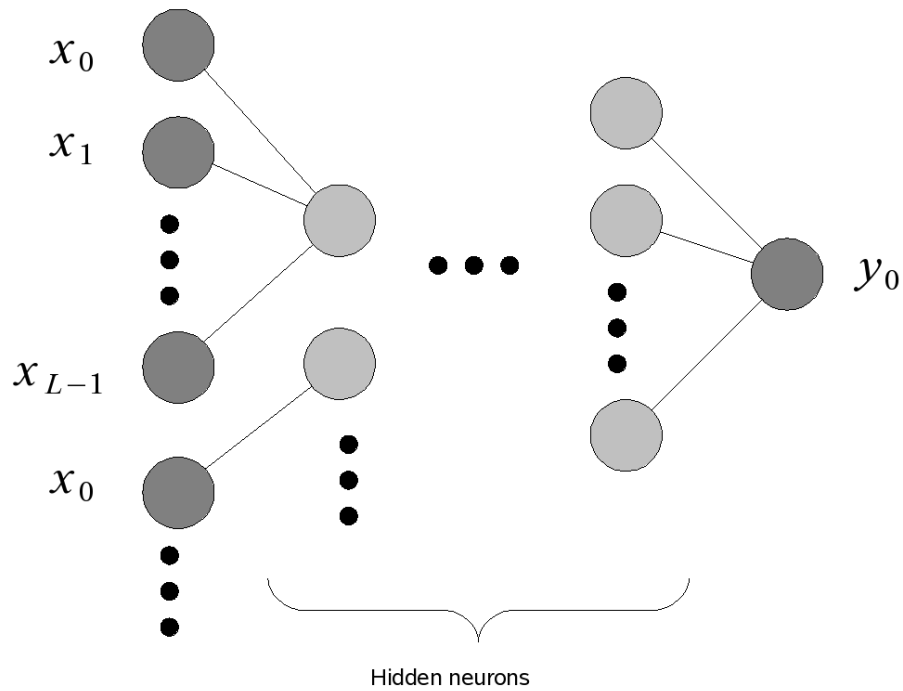


Figure 6: The Volterra network.

2 Number of operations for calculating output values

2.1 Activation functions of neurons

An artificial neuron model is an individual identity element which must sum up signals which enter with associated weight coefficients (this is an analog of the biological prototype of the synaptic force). Afterwards an activation function (which is usually nonlinear) is executed and formulates the output signal of the neuron.

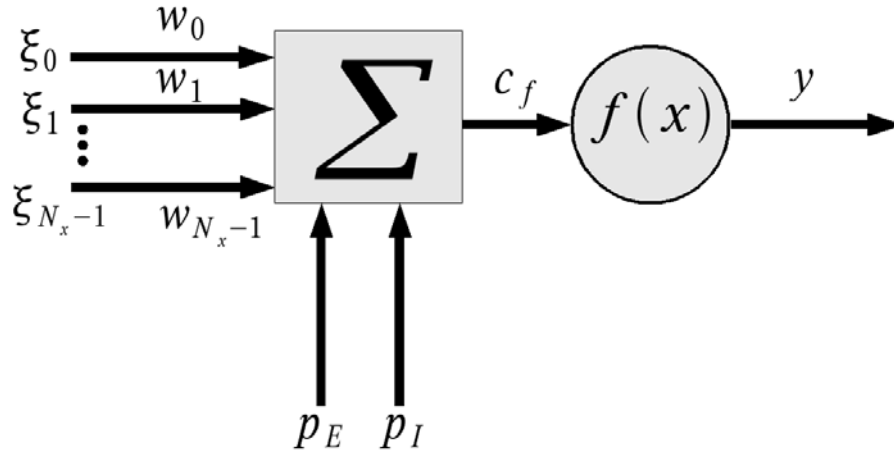


Figure 7: The model of the McCulloch-Pitts neuron.

The model of McCulloch-Pitts was one of the first models of the artificial neuron. For the signal generation it uses the threshold function (3).

$$y = \begin{cases} 1, & x_f > 0; \\ 0, & x_f \leq 0; \end{cases} \quad (3)$$

The function (3) is an activation function. The argument of this function is calculated by formula (4).

$$x_f = \sum_{i=0}^{N_x-1} \xi_i w_i \quad (4)$$

where \vec{w} is the weight coefficients vector, $\vec{\xi}$ is the vector of input signals which has length N_x .

The McCulloch-Pitts model with its training strategy is called “the simple perceptron” [16]. There are other activation function flavors. The functions used most often are the sigmoid and the tangent.

As it is shown in figure 7 a general neuron has two impulses (adjugate weight coefficients). One of these impulses (which is called “the internal impulse” (p_I)) is fixed when weight coefficients are trained and the second impulse (which is called “the external impulse” (p_E)) is updated together with the former. The activation function argument has a coefficient and so the neuron output value can be calculated by formula (5).

$$y = f \left(c_f \left(\sum_{i=0}^{N_x-1} (\xi_i w_i) + p_I + p_E \right) \right) \quad (5)$$

The usage of impulses (p_I and p_E) and a coefficient (c_f) allows to customize activation functions flexibly.

It symbolizes a number of multiplication operations which are executed for the calculation of the i -th neuron output value (by formula (5)) as ζ_n , and it symbolizes a addition operations number as $\hat{\zeta}_n$. Values ζ_n and $\hat{\zeta}_n$ of different activation functions are showed in table 1.

Table 1: Numbers of operations which are executed for calculation of output neuron value.

The function flavor	The number of addition operations ($\hat{\zeta}_n$)	The number of multiplication operations (ζ_n)
$y = c_f x_f$	2	1
$y = c_f^2 x_f^2$	2	2
$y = c_f^3 x_f^3$	2	3
$y = c_f^4 x_f^4$	2	3
$y = \sin(c_f x_f)$	2	28
$y = \cos(c_f x_f)$	2	28
$y = \tan(c_f x_f)$	2	24
$y = \text{ctg}(c_f x_f)$	3	24
$y = \frac{1}{1+c_f x_f}$	3	2
$y = \frac{1}{1+e^{c_f x_f}}$	3	12
$y = \frac{1}{1+e^{ c_f x_f }}$	3	12

2.2 Calculation of output values of different network types

The number of multiplication operations which are necessary for the calculation of the ANN output values depends on the number of multiplication operations which are executed for the calculation of output values of neurons and the number of synapses:

$$\zeta_y = \sum_{i=0}^{l_\mu-1} \zeta_n(\mu_i) + l_w \quad (6)$$

where l_μ is the number of neurons in the ANN, l_w is the number of synapses (weights). Analogously the number of addition operations which are necessary for the calculation of ANN output values is calculated:

$$\hat{\zeta}_y = \sum_{i=0}^{l_\mu-1} \hat{\zeta}_N(\mu_i) + l_w + \hat{\zeta}_I \quad (7)$$

where $\hat{\zeta}_I$ is the number of addition operations which are necessary for changing the marker of the obstruction cocycle.

It is possible to modify formulas (6)-(7) for different types of ANNs. So formulas (8)-(9) formulate operations numbers for the MLP:

$$\zeta_{yM} = \sum_{i=0}^{l_\mu-1} \zeta_n(\mu_i) + \sum_{i=1}^{N_L-1} \hat{N}_i \hat{N}_{i-1} \quad (8)$$

$$\hat{\zeta}_{yM} = \sum_{i=0}^{l_\mu-1} \hat{\zeta}_n(\mu_i) + 2 \sum_{i=1}^{N_L-1} \hat{N}_i \hat{N}_{i-1} + 2N_L + P + L \quad (9)$$

In a similar way one can calculate numbers of operations for the cascade-correlation network (calculated by formulas (10) and (11)) and the Volterra network (calculated by formulas (12) and (13)):

$$\zeta_{yC} = \sum_{i=0}^{l_{\mu}-1} \zeta_n(\mu_i) + \sum_{i=0}^{\hat{N}_1-1} (\hat{N}_0 + i) + P(\hat{N}_0 + \hat{N}_1) \quad (10)$$

$$\hat{\zeta}_{yC} = \sum_{i=0}^{l_{\mu}-1} \hat{\zeta}_n(\mu_i) + 2 \sum_{i=0}^{\hat{N}_1-1} (\hat{N}_0 + i) + 2P(\hat{N}_0 + \hat{N}_1) + \hat{N}_1 + 2P + L \quad (11)$$

$$\zeta_{yV} = \sum_{i=0}^{l_{\mu}-1} \zeta_n(\mu_i) \quad (12)$$

$$\hat{\zeta}_{yV} = \sum_{i=0}^{l_{\mu}-1} \hat{\zeta}_n(\mu_i) + 2 \sum_{i=1}^{N_L-1} \hat{N}_i \hat{N}_{i-1} + L(N_L - 1) \quad (13)$$

2.3 Reduction of addition operations to multiplication operations

For reduction of addition operations to multiplication operations the coefficient σ is used. This coefficient value is directly proportional to the time spent for one addition operation and inversely related to the time spent for one multiplication operation. So one multiplication operation needs the time which is necessary for σ addition operations and one addition operation can be changed to σ multiplication operations. So it is possible to formulate equations (14)-(16).

$$\begin{aligned} z_{yM} = \zeta_{yM} + \sigma \hat{\zeta}_{yM} = & \sum_{i=0}^{l_{\mu}-1} \zeta_n(\mu_i) + \sum_{i=1}^{N_L-1} \hat{N}_i \hat{N}_{i-1} + \\ & + \sigma \left(\sum_{i=0}^{l_{\mu}-1} \hat{\zeta}_N(\mu_i) + 2 \sum_{i=1}^{N_L-1} \hat{N}_i \hat{N}_{i-1} + 2N_L + P + L \right) \end{aligned} \quad (14)$$

$$\begin{aligned} z_{yC} = \zeta_{yC} + \sigma \hat{\zeta}_{yC} = & \sum_{i=0}^{l_{\mu}-1} \zeta_n(\mu_i) + \sum_{i=0}^{\hat{N}_1-1} (\hat{N}_0 + i) + \\ & + P(\hat{N}_0 + \hat{N}_1) + \sigma \left(\sum_{i=0}^{l_{\mu}-1} \hat{\zeta}_N(\mu_i) + 2 \sum_{i=0}^{\hat{N}_1-1} (\hat{N}_0 + i) \right) + \\ & + \sigma \left(2P(\hat{N}_0 + \hat{N}_1) + \hat{N}_1 + 2P + L \right) \end{aligned} \quad (15)$$

$$\begin{aligned} z_{yV} = \zeta_{yV} + \sigma \hat{\zeta}_{yV} = & \sum_{i=0}^{l_{\mu}-1} \zeta_n(\mu_i) + \\ & + \sigma \left(\sum_{i=0}^{l_{\mu}-1} \hat{\zeta}_N(\mu_i) + 2 \sum_{i=1}^{N_L-1} \hat{N}_i \hat{N}_{i-1} + L(N_L - 1) \right) \end{aligned} \quad (16)$$

3 Number of operations of algorithms of training

3.1 The general coefficient of the efficiency

The coefficient of efficiency depends on the numbers of operations which are calculated by the serial and the parallel algorithm. Its value is calculated by formula (17):

$$\alpha(Z) = \frac{z}{nZ} \quad (17)$$

where z is the number of operations in the serial algorithm, Z is the number of operations in the parallel algorithm (in the processor which executes the maximum number of operations) and n is the number of processors.

3.2 Level of calculating the value of the target function

The value of the target function is calculated by formula (18). In this equation the calculation of ε for the i -th row of the pattern needs $\zeta_y + 1$ multiplication operations (ζ_y operations for the calculation of ANN output values and one operation for the squaring) and $\hat{\zeta}_y + 2P$ addition operations.

$$\varepsilon = \sum_{i=0}^{N-1} \sum_{j=0}^{P-1} (d_{i,j} - F(\vec{x}_i, \vec{w}, \vec{\mu})_j)^2 \quad (18)$$

where x and d are patterns (input and output) which consist of N rows, \vec{w} is the weights coefficients vector, $\vec{\mu}$ is the vector of activation functions of neurons, F is the function which calculates output values of the ANN, P is the number of output neurons in the ANN.

So for the calculation of the full inaccuracy ε $N(\zeta_y + 1) + 1$ multiplication and $\hat{\zeta}_y + 2P$ addition operations are needed. And it is necessary to execute N addition operations for the summation.

So it is possible to formulate equations (19)-(21) which show the numbers of operations which are executed by the algorithms of calculating the value of the target function.

$$\zeta_\varepsilon = N(\zeta_y + 1) + 1 \quad (19)$$

$$\hat{\zeta}_\varepsilon = N(\hat{\zeta}_y + 2P) \quad (20)$$

$$z_\varepsilon = \zeta_\varepsilon + \sigma \hat{\zeta}_\varepsilon = N(\zeta_y + 1 + \sigma \hat{\zeta}_y + 2\sigma P) + 1 \quad (21)$$

If there are n processors and the pattern consists of N rows then each processor can calculate the inaccuracy for the part of the pattern (equation (23)) for processor number k and equation (22) for the lead processor. Then the zero processor sums up (formula (24)). So the lead processor sends M rows to each processor, calculates the inaccuracy by \hat{M} rows, receives the inaccuracies from other processors and calculates

the result. Values of M and \hat{M} are calculated by formulas (25)-(26) [4, 17].

$$\varepsilon_0 = \sum_{i=0}^{\hat{M}-1} \sum_{j=0}^{P-1} (d_{i,j} - F(\vec{x}_i, \vec{w}, \vec{\mu})_j)^2 \quad (22)$$

$$\varepsilon_k = \sum_{i=0}^M \sum_{j=0}^{P-1} (d_{i+(k*M)+\hat{M},j} - F(\vec{x}_{i+(k*M)+\hat{M}}, \vec{w}, \vec{\mu})_j)^2, \quad k > 0 \quad (23)$$

$$\varepsilon = \frac{1}{N} \sum_{k=0}^{n-1} \varepsilon_k \quad (24)$$

$$M = \begin{cases} \lfloor \frac{N}{n} \rfloor, & N \bmod n = 0; \\ \lfloor \frac{N}{n-1} \rfloor, & N \bmod n \neq 0; \end{cases} \quad (25)$$

$$\hat{M} = \begin{cases} M, & N \bmod n = 0; \\ N - M(n-1), & N \bmod n \neq 0; \end{cases} \quad (26)$$

It is necessary for the parallel algorithm:

- to divide the pattern into n parts and send it to processors before the training;
- to send the vector of weights coefficients \vec{w} to each nonzero processor and receive the inaccuracy value ε_k in each iteration of the calculation of the value of the target function.

For sending or receiving one element one multiplication and two addition operations are necessary. So for sending the pattern which will be worked on by the k -th processor the lead processor executes $M(P+L)$ multiplications and $2M(P+L)$ addition operations and it executes $M(P+L)(n-1)$ multiplications and $2M(P+L)(n-1)$ addition operations for sending patterns to all nonzero processors. Thus the number of pattern elements which are sent is \hat{P} ($\hat{P} = MP + ML$). The segmentation of the pattern into n parts needs two multiplication and N addition operations and thus for accomplishing the first step in the lead processor $2 + \hat{P}(n-1)$ multiplication and $N + 2\hat{P}(n-1)$ addition operations and \hat{P} multiplication and $2\hat{P}$ addition operations in nonzero processors are needed. The non-lead processors cannot begin to receive data until the lead processor sends it so k -th processor ($k > 0$) waits $2 + k\hat{P}$ multiplication and $N + 2k\hat{P}$ addition operations which are executed by the lead processor for sending patterns to the k -th processor. Thus the k -th processor ($k > 0$) executes $2 + k\hat{P} + \sigma N + 2\sigma k\hat{P}$ vacuous operations which conform with the preparatory operations in the lead processor and $\gamma(\hat{P}, v)$ operations of the sending (the time of making these operations conforms with the time of the interprocessor sending \hat{P} numbers where the time rate of the interconnection is v). So the lead processor executes $C_{\varepsilon_0}^{(0)}$ multiplication operations, and the k -th nonzero processor executes $C_{\varepsilon_k}^{(0)}$. The values of $C_{\varepsilon_0}^{(0)}$ and $C_{\varepsilon_k}^{(0)}$ are calculated by formulas (27)–(28)).

$$C_{\varepsilon_0}^{(0)} = 2 + \hat{P}(n-1) + \sigma N + 2\hat{P}\sigma(n-1) \quad (27)$$

$$C_{\varepsilon_k}^{(0)} = 2 + k\hat{P} + \sigma N + 2\sigma k\hat{P} + \gamma(\hat{P}, v) + \hat{P} + 2\sigma\hat{P} \quad (28)$$

As figure 8 shows, the parallel calculation of the inaccuracy consists of few steps:

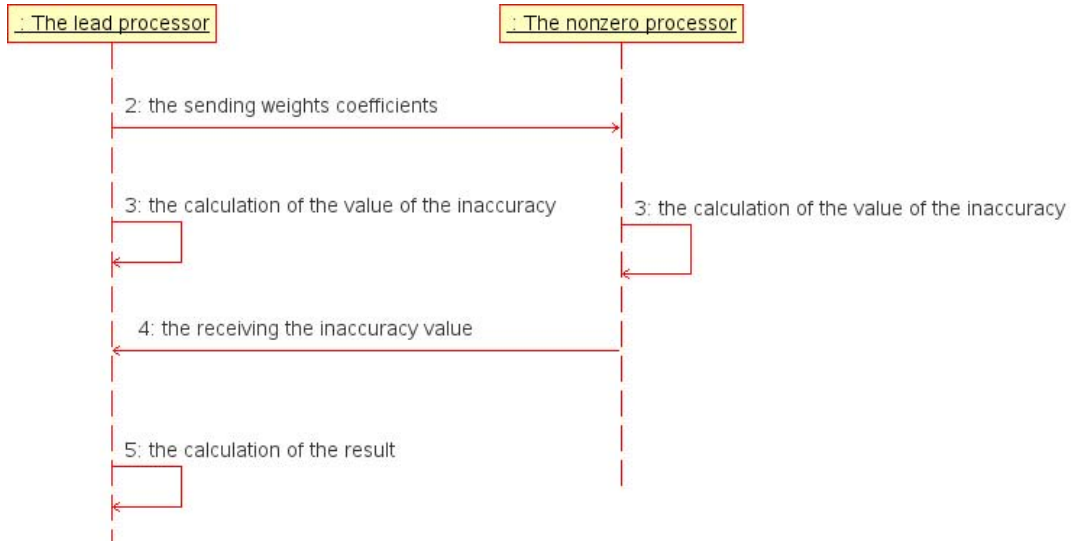


Figure 8: The diagram of steps of the parallel calculation of the target function value.

1. sending elements of the vector of weight coefficients (the length of this vector is l_w) to all nonzero processors. For this the lead processor executes $l_w(n-1)$ multiplication and $2l_w(n-1)$ addition operations and the other processors execute l_w multiplication and $2l_w$ addition operations for receiving the data. So the k -th processor executes $C_{\varepsilon k}^{(0)}$ and the lead processor executes $C_{\varepsilon 0}^{(1)}$ operations. The values of $C_{\varepsilon 0}^{(0)}$ and $C_{\varepsilon k}^{(1)}$ are calculated by formulas (29)–(30).

$$C_{\varepsilon 0}^{(1)} = (n-1)(l_w + 2\sigma l_w) \quad (29)$$

$$C_{\varepsilon k}^{(1)} = kl_w + 2\sigma kl_w + \gamma(l_w, v) + l_w + 2\sigma l_w \quad (30)$$

2. calculating the value of the inaccuracy: each processor uses its part of the pattern, so the numbers of operations are calculated by formula (31) for the lead processor and (32) for the nonzero processors.

$$C_{\varepsilon 0}^{(2)} = \hat{M}(\zeta_y + 1 + \sigma \hat{\zeta}_y + 2\sigma P) + 1 \quad (31)$$

$$C_{\varepsilon k}^{(2)} = M(\zeta_y + 1 + \sigma \hat{\zeta}_y + 2\sigma P) + 1 \quad (32)$$

3. receiving the inaccuracy value by the lead processor. For this, the nonzero processors execute one multiplication and two addition operations, and the zero processor executes $n-1$ multiplication and $2n-2$ addition operations for receiving and $1+2\sigma+\gamma(1, v)$ operations for waiting. So the lead processor executes $C_{\varepsilon 0}^{(3)}$ operations (the value of $C_{\varepsilon 0}^{(3)}$ is calculated by formula (33)) and other processors execute $C_{\varepsilon k}^{(3)}$ (this value is calculated by formula (34)).

$$C_{\varepsilon 0}^{(3)} = n-1 + 2\sigma(n-1) + \gamma(1, v) + 1 + 2\sigma \quad (33)$$

$$C_{\varepsilon k}^{(3)} = 1 + 2\sigma \quad (34)$$

4. calculating the result by the zero processor. For this, it is necessary to execute $n-1$ addition operations and one multiplication operation.

So the number of operations which are necessary for these steps are shown in table 2.

Table 2: Numbers of operations which are necessary for the steps of the parallel calculation of the inaccuracy.

Step	Zero processor	Nonzero (k -th) processor
1	$l_w(n-1)(1+2\sigma)$	$kl_w + 2\sigma kl_w + \gamma(l_w, v) + l_w + 2\sigma l_w$
2	$\hat{M}(z_y + 1 + 2\sigma P) + 1$	$M(z_y + 1 + 2\sigma P) + 1$
3	$n + 2\sigma n + \gamma(1, v)$	$1 + 2\sigma$
4	$\sigma(n-1) + 1$	

Before the lead processor begins to receive the results of the operations which were calculated by other processors it sends weight coefficients to all other processors and calculates ε_0 (for this it executes C_{ε_0} operations) and the other processors receive the vector \vec{w} , calculate the value of the inaccuracy ε_k and send the result to the lead processor.

$$\hat{C}_{\varepsilon_0} = l_w(n-1)(1+2\sigma) + \hat{M}(\zeta_y + 1 + \sigma\hat{\zeta}_y + 2\sigma P) + 1 \quad (35)$$

$$\hat{C}_{\varepsilon_k} = (1+2\sigma)(kl_w + l_w + 1) + \gamma(l_w, v) + M(\zeta_y + 1 + \sigma\hat{\zeta}_y + 2\sigma P) + 1 \quad (36)$$

It is necessary to consider the time of sending the inaccuracy value ($\gamma(1, v)$). Thus the zero processor can begin to receive data from the k -th processor after $\max(\hat{C}_{\varepsilon_0}, \hat{C}_{\varepsilon_k} + \gamma(1, v))$ operations. The receiving finishes after the slowest processor has sent its inaccuracy value, and this is why the parallel calculation of the inaccuracy needs Z_ε operations (calculated by formula (37)).

$$Z_\varepsilon = \max_{k=1..n-1} \left(\max(\hat{C}_{\varepsilon_0} + k(2\sigma + 1), \hat{C}_{\varepsilon_k} + \gamma(1, v)) \right) + \sigma n - 1\sigma + 1 \quad (37)$$

So it is possible to conclude (38) which shows the efficiency of the algorithm which executes the parallel calculation of the value of the target function.

$$\alpha_\varepsilon(Z) = \frac{I_\varepsilon Z_\varepsilon + \lambda_\varepsilon}{nI_\varepsilon Z_\varepsilon + C_{\varepsilon k}^{(0)} + \lambda_\varepsilon} \quad (38)$$

where I_ε is the number of operations for the calculation of the inaccuracy, λ_ε is the number of other operations of algorithms (which do not belong to the calculation of the inaccuracy).

3.3 Full enumeration method

This method searches all variants of weight coefficients values. So the value of the i -th weight coefficient in the I -th iteration is calculated by formula (39). The method

iterates I_F times (the value of I_F is calculated by formula (41)).

$$w_i^{(I)} = l_0^{down} + s_i \times \left(\left[\frac{I}{\hat{s}(i)} \right] \bmod \left(\left[\frac{l_i^{up} - l_i^{down}}{s_i} \right] + 1 \right) \right) \quad (39)$$

$$\hat{s}(i) = \begin{cases} 1, & i = l_w - 1; \\ \prod_{j=i+1}^{l_w-1} \left(\left[\frac{l_j^{up} - l_j^{down}}{s_j} \right] + 1 \right), & i < l_w - 1; \end{cases} \quad (40)$$

$$I_F = \prod_{i=0}^{l_w-1} \left(\left[\frac{l_i^{up} - l_i^{down}}{s_i} \right] + 1 \right) \quad (41)$$

where l_i^{up} , l_i^{down} are the upper and lower limits of the i -th weight coefficient, s_i is the step of the i -th scanning of the weight coefficient, and l_w is the weights count.

It is possible to calculate the number of multiplication operations which are executed by the serial algorithm of the full enumeration by formula (42).

$$z_{wF} = z_\varepsilon I_F + 2\sigma I_F \quad (42)$$

where z_ε is the number of operations which are needed for the calculation of the inaccuracy value.

If there are n processors then each nonzero processor iterates J_F times (formula (43)), and the lead processor iterates \hat{J}_F times (formula (44)) [17].

$$J_F = \begin{cases} \left[\frac{I_F}{n} \right], & I_F \bmod n = 0; \\ \left[\frac{I_F}{n-1} \right], & I_F \bmod n \neq 0; \end{cases} \quad (43)$$

$$\hat{J}_F = \begin{cases} J_F, & I_F \bmod n = 0; \\ I_F - J_F(n-1), & I_F \bmod n \neq 0; \end{cases} \quad (44)$$

For the calculation of the number of multiplication operations which are executed by the parallel algorithm of the full enumeration it is necessary to analyse the steps of this algorithm. As figure 9 shows, the parallel algorithm consists of five steps:

1. initialization;
2. sending data from the lead processor to other processors;
3. enumeration of values of weight coefficients which belong to this processor;
4. sending data from all processors to zero processor;
5. selection of the best configuration;

In the first step the lead processor executes $2l_w$ addition operations (l_w for the organization of the obstruction cocycle and l_w for the awarding). So $C_{wF0}^{(1)} = 2\sigma l_w$ and $C_{wFk}^{(1)} = 0$. In the second step the lead processor executes $l_w(n-1)$ multiplication and $2l_w(n-1)$ addition operations. The other processors execute l_w multiplication and $2l_w$ addition operations but they can begin before the zero processor sends data, so the nonzero processors execute $C_{Fk}^{(2)}$ multiplication operations (calculated by formula (45)).

$$C_{wFk}^{(2)} = l_w(k + 2k\sigma + 2\sigma + 1) + \gamma(l_w, v) \quad (45)$$

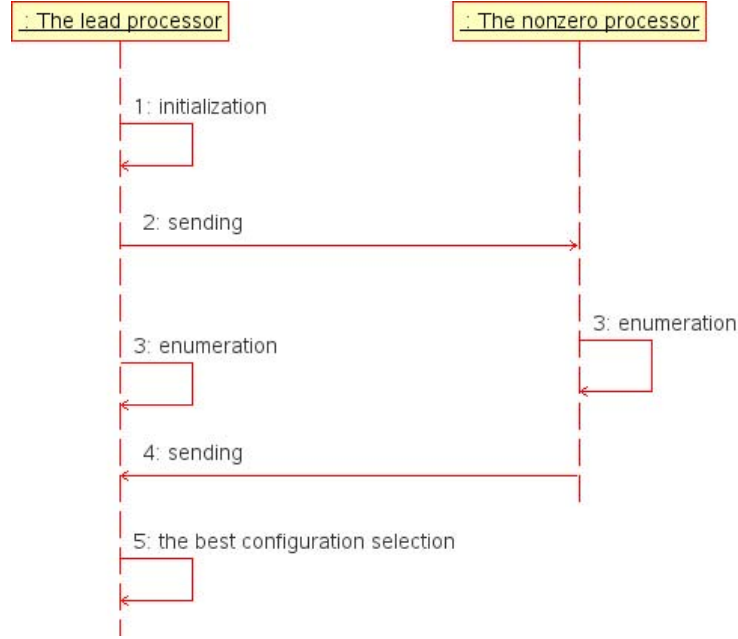


Figure 9: The diagram of the parallel full enumeration method steps.

Table 3: Numbers of multiplication operations in steps of the parallel full enumeration algorithms.

Step	Zero processors	Nonzero (k -th) processor
1	$2\sigma l_w$	
2	$l_w(n-1)(1+2\sigma)$	$l_w(k+2k\sigma+2\sigma+1)+\gamma(l_w, v)$
3	$z_\varepsilon \hat{J}_F + 2\sigma \hat{J}_F$	$z_\varepsilon J_F + 2\sigma J_F$
4	$n(l_w+1)(1+2\sigma)+\gamma(l_w+1)$	$(l_w+1)(1+2\sigma)$
5	$2\sigma n$	

The number of operations in the third step is analogous to the number of operations which are executed by the serial algorithm.

In the fourth step the lead processor needs to get the best weights coefficients and the inaccuracy value which belongs to these coefficients. It needs to get this information from each processor, so it executes $(n-1)(l_w+1)$ multiplication and $2(n-1)(l_w+1)$ addition operations, and the nonzero processors execute l_w+1 multiplication and $2l_w+2$ addition operations. And the lead processor has to wait for the sending.

In the fifth step zero processor executes $2n$ addition operations. The reduction of addition operations to multiplication operations allows to get values which are shown in table 3.

Before the the lead processor begins to receive weight coefficients and inaccuracy values, it executes three steps (the total number of operations is \hat{C}_{F0}), and the other processors execute four steps (the total number of operations is \hat{C}_{Fk}). The alues of \hat{C}_{F0}

and \hat{C}_{Fk} are calculated by formulas (46)–(47).

$$\hat{C}_{wF0} = 2\sigma l_w + l_w(n-1)(1+2\sigma) + z_\varepsilon \hat{J}_F + 2\sigma \hat{J}_F \quad (46)$$

$$\begin{aligned} \hat{C}_{wFk} &= l_w(k+2k\sigma+\sigma+1) + \gamma(l_w, v) + z_\varepsilon J_F + 2\sigma J_F + \\ &+ (l_w+1)(1+2\sigma) \end{aligned} \quad (47)$$

And it is necessary to consider the time for sending the inaccuracy value and the weight coefficients. The receiving finishes after the the slowest processor has sent the inaccuracy, and this is why the parallel calculation of inaccuracy needs to execute Z_{wF} operations. This value is calculated by formula (48).

$$\begin{aligned} Z_{wF} &= \max_{k=1..n-1} \left(\max \left(\hat{C}_{wF0} + k(l_w+1)(2\sigma+1), \hat{C}_{wFk} + \gamma(l_w+1, v) \right) \right) + \\ &+ (l_w+1)(1+2\sigma) + 2\sigma n \end{aligned} \quad (48)$$

So the efficiency of the parallel algorithm can be defined in formula (49):

$$\alpha_{wF}(Z) = \frac{z_{wF}}{nZ_{wF}} = \frac{z_\varepsilon I_F + 2\sigma I_F}{nZ_{wF}} \quad (49)$$

3.4 Gradient methods

Gradient methods are based on the calculation of a gradient vector and change the values of the weight coefficients in the opposite direction:

$$\vec{w}^{(I)} = \vec{w}^{(I-1)} - \vec{s}^{(I)} \nabla \varepsilon^{(I)} \quad (50)$$

where \vec{s} is the coefficient of training and $\nabla \varepsilon$ is the gradient which is calculated by formula (51) [18, 19]. The values of $\vec{p}(\vec{w})$ and the coefficient vector \vec{s} are selected and it is necessary for $\varepsilon^{(I)}$ to be less than $\varepsilon^{(I-1)}$ for each element of the coefficient vector (as is shown in formula (52)). The searching of the minimum continues until the reduction of the gradient norm is below the value which was set before (or the finishing the time which is allowed for the training).

$$\nabla \varepsilon = \left(\frac{\partial \varepsilon}{\partial w_0}, \frac{\partial \varepsilon}{\partial w_1}, \dots, \frac{\partial \varepsilon}{\partial w_{l_w-1}} \right) \quad (51)$$

$$\vec{w}^{(I)} = \vec{w}^{(I-1)} + \vec{s}^{(I)} \vec{p}(\vec{w}^{(I-1)}) \quad (52)$$

Three methods which are the most useful will be considered. These are the steepest descent method, QuickProp and RPROP. They use different formulas for updating the weight coefficients.

The steepest descent method uses formula (53) or sometimes (54).

$$\vec{p}(\vec{w}^{(I)}) = -\nabla \varepsilon^{(I-1)} \quad (53)$$

$$w_i^{(I)} = w_i^{(I-1)} + s_i^{(I)} \frac{\partial \varepsilon(\vec{w}^{(I-1)})}{\partial w_i^{(I-1)}} + q \Delta w_i^{(I-1)} \quad (54)$$

Table 4: Numbers of operations which are necessary for one iteration of gradient methods.

Method	The multiplicative operations number	The additive operations number
The steepest descent	$l_w(\zeta_\varepsilon + 4)$	$l_w(\hat{\zeta}_\varepsilon + 5)$
QuickProp	$l_w(\zeta_\varepsilon + 4)$	$l_w(\hat{\zeta}_\varepsilon + 5)$
RPROP	$l_w(\zeta_\varepsilon + 4)$	$l_w(\hat{\zeta}_\varepsilon + 4)$

where q is the coefficient of the moment which lies in the interval $[0, 1]$. As can be seen, this algorithm executes three multiplication and two additive operations for the calculation of updating one element of the weight coefficient vector.

The algorithm QuickProp which was suggested by Falman uses formula (55) [20, 21].

$$\Delta w_i^{(I)} = \begin{cases} q_i^{(I)} \Delta w_i^{(I-1)}, & \Delta w_i^{(I-1)} \neq 0; \\ s_i^{(I)} \nabla \varepsilon_i^{(I-1)}, & \Delta w_i^{(I-1)} = 0; \end{cases} \quad (55)$$

$$q_i^{(I)} = \min \left(\frac{\nabla \varepsilon_i^{(I-1)}}{\nabla \varepsilon_i^{(I-2)} - \nabla \varepsilon_i^{(I-1)}}, q_{MAX} \right) \quad (56)$$

The maximal value of the coefficient of the moment q_{MAX} is 1.75 [22]. This algorithm executes three multiplicative and two additive operations too.

Another heuristic algorithm which is called RPROP (Resilient back PROPagation) was developed by Riedmiller and Brawn [23, 24]. It calculates the elements of the vector of coefficients by formula (57).

$$s_i^{(I)} = \begin{cases} \min(q_a s_i^{(I-1)}, q_{MAX}), & \nabla \varepsilon_i^{(I-1)} \nabla \varepsilon_i^{(I-2)} > 0 \\ \max(q_b s_i^{(I-1)}, q_{MIN}), & \nabla \varepsilon_i^{(I-1)} \nabla \varepsilon_i^{(I-2)} < 0 \\ s_i^{(I-1)}, & \nabla \varepsilon_i^{(I-1)} \nabla \varepsilon_i^{(I-2)} = 0 \end{cases} \quad (57)$$

where q_{MAX} and q_{MIN} are maximal and minimal values of the coefficient of the training ($q_{MAX}=50$, $q_{MIN} = 10^{-6}$ [24]), q_a and q_b are constant ($q_a=1.2$, $q_b=0.5$) [25]. The speciality of this method is that it ignores the gradient value. The vector of the direction $\vec{p}(\vec{w})$ is determined by the sign $(-1, 0, 1)$ [26]. This algorithm executes three multiplication and one addition operations.

It is necessary to execute $\xi_\varepsilon + 1$ multiplicative and $\xi_\varepsilon + 3$ additive operations for the calculation of one element of the gradient. So it is possible to formulate that one operation needs the operations numbers which are showed in table 4.

As shown in figure 10, each parallel iteration of the gradient method consist of few steps:

- sending the vector of weight coefficients \vec{w} to all nonzero processors by the lead processor;

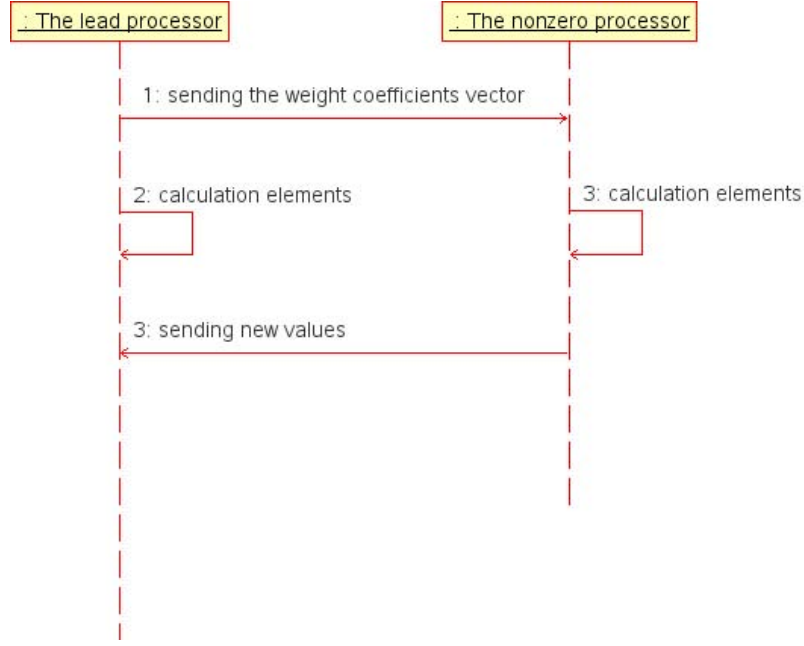


Figure 10: The diagram of the parallel gradient method steps.

- calculating the elements of the part of weight coefficients and the gradient which belongs to the current processor;
- sending new values of weight coefficients to the lead processor by all nonzero processors.

The first step needs $l_w(n - 1)$ multiplicative and $2l_w(n - 1)$ additive operations in the lead processor and $l_w(k + 1) + \gamma(l_w, v)$ operations in the k -th processor (k). The number of operations which are necessary for the second step are shown in table 5. For the third step it executes \hat{l}_w multiplicative and $2\hat{l}_w$ additive operations in the non-lead processors and it executes $\hat{l}_w n + \gamma(l_w, v)$ ($\hat{l}_w(n - 1)$ multiplicative ($\hat{l}_w(n - 1)$ operations for the receiving and \hat{l}_w operations for the waiting) and $2\hat{l}_w(n - 1)$ additive operations in the lead processor.

The first step needs $C_{wG0}^{(1)}$ operations in the lead processor and $C_{wGk}^{(1)}$ in other processors (these values are calculated by formulas (58)-(59)). The second step needs $C_{wG0}^{(2)}$ and $C_{wGk}^{(2)}$ operations (see the data from table 5) and the third step needs $C_{wGk}^{(3)}$ operations (calculated by formula (60)) in the nonzero processors. So the lead processor has to wait for \hat{C}_{wG} operations before it starts receiving. The value of \hat{C}_{wG} is calculated by formula (61).

$$C_{wG0}^{(1)} = l_w(n - 1)(1 + 2\sigma) \quad (58)$$

$$C_{wGk}^{(1)} = l_w(k + 1)(1 + 2\sigma) + \gamma(l_w, v) \quad (59)$$

$$C_{wGk}^{(3)} = \hat{l}_w + 2\sigma\hat{l}_w \quad (60)$$

$$\hat{C}_{wG} = \max \left(C_{wG0}^{(1)} + C_{wG0}^{(2)}, C_{wGk}^{(1)} + C_{wGk}^{(2)} + C_{wGk}^{(3)} + \gamma(\hat{l}_w, v) \right) \quad (61)$$

So the parallel algorithm executes Z_{wG} operations (calculated by formula (62)), and the efficiency of parallel gradient algorithms which is calculated by formula (63) may

Table 5: Numbers of operations which are necessary for the second step of the parallel iteration of gradient methods.

Method	Number of mult. operations in the lead processor	Number of additive operations in the lead processor	Number of mult. operations in the non-lead processor	Number of additive operations in the non-lead processor
The steepest descent	$\hat{N}_w (\zeta_\varepsilon + 4)$	$\hat{N}_w (\hat{\zeta}_\varepsilon + 5)$	$\hat{l}_w (\zeta_\varepsilon + 4)$	$\hat{l}_w (\hat{\zeta}_\varepsilon + 5)$
QuickProp	$\hat{N}_w (\zeta_\varepsilon + 4)$	$\hat{N}_w (\hat{\zeta}_\varepsilon + 5)$	$\hat{l}_w (\zeta_\varepsilon + 4)$	$\hat{l}_w (\hat{\zeta}_\varepsilon + 5)$
RPROP	$\hat{N}_w (\zeta_\varepsilon + 4)$	$\hat{N}_w (\hat{\zeta}_\varepsilon + 4)$	$\hat{l}_w (\zeta_\varepsilon + 4)$	$\hat{l}_w (\hat{\zeta}_\varepsilon + 4)$

be formulated as (64) for the method of the steepest descent and QuickProp and (65) for RPROP.

$$Z_{wG} = I_G(\hat{C}_{wG} + k(\hat{l}_w + 1)(2\sigma + 1) + 2\sigma\hat{l}_w(n - 1)) + \lambda_G \quad (62)$$

$$\alpha_{wG}(Z) = \frac{z_{wG}}{nZ_{wG}} \quad (63)$$

$$\alpha_{wGQ}(Z) = \frac{I_G l_w (\zeta_\varepsilon + 4 + \sigma \hat{\zeta}_\varepsilon + 5\sigma) + \lambda_G}{nZ_{wG}} \quad (64)$$

$$\alpha_{wGR}(Z) = \frac{I_G l_w (\zeta_\varepsilon + 4 + \sigma \hat{\zeta}_\varepsilon + 4\sigma) + \lambda_G}{nZ_{wG}} \quad (65)$$

where I_G is the number of iterations and, λ_G is the number of other operations of the algorithm.

4 Results

4.1 Experiments

For checking the formulas which were developed here an experiment was done. In this experiment a forecast of currency exchange rates €/\$ was attempted at. It uses two ANN structures, namely a multilayer perceptron (Fig. 11) and a Volterra network. The pattern consisted of 300 rows and each row consisted of 12 input and one output values. The results of this experiment are shown in table 6.

Another experiment was the prediction of shrimp mass caught in the Indian ocean. For this experiment the MLP (Fig. 12) was used. Results are shown in table 7.

Thus the efficiency of the parallel algorithm of the full enumeration is about 95%, the efficiency of the gradient algorithms is about 93% and the efficiency of the parallel algorithms of the target function minimizing is about 91%.

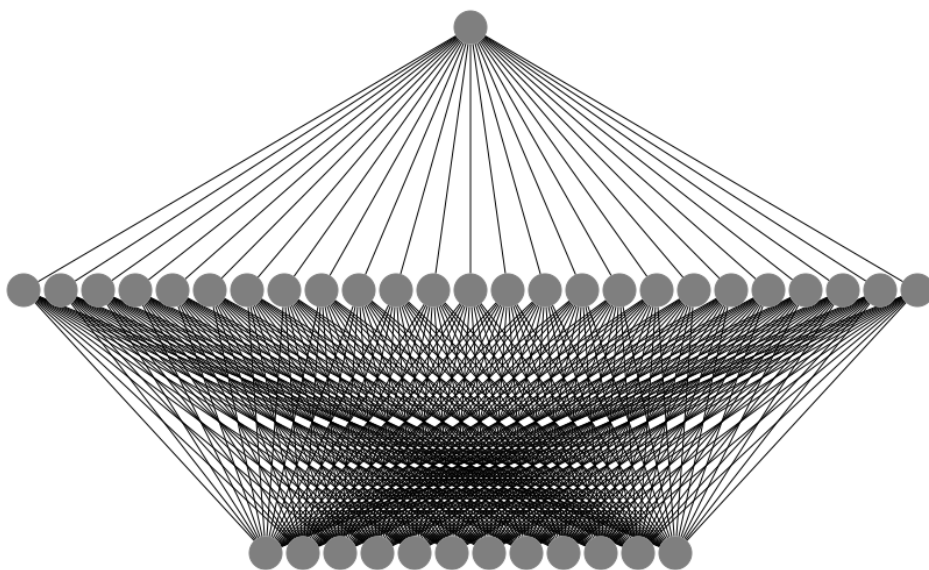


Figure 11: The MLP which was used for the prediction of currency pair quotations.

Table 6: Values of efficiency coefficient for prediction of currency pair quotations.

4	0.9127	0.9521	0.9319
6	0.9111	0.9509	0.9302
8	0.9101	0.9492	0.9293

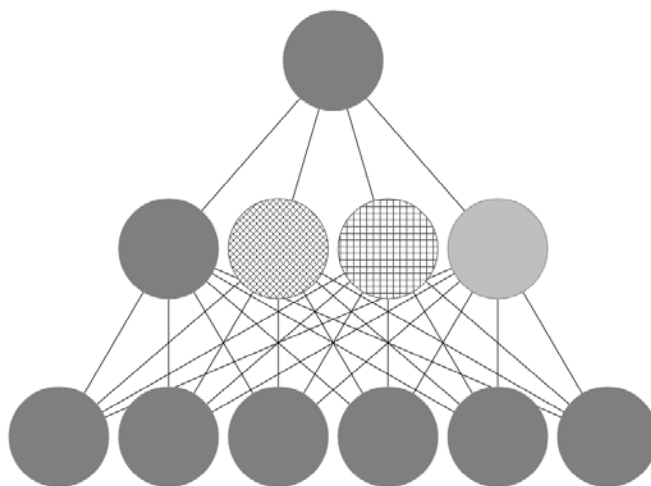


Figure 12: The MLP which was used for the forecast of the shrimp mass which was caught.

4.2 Conclusion

The time which is necessary for training the ANN by the parallel algorithm can be calculated by formula (66):

$$\tau \approx \frac{t}{n\alpha(Z)} \quad (66)$$

Table 7: Values of the efficiency coefficient for predicting the shrimp mass which was caught.

Processors number	α_ε	α_{wF}	α_{wG}
4	0.9124	0.9518	0.9312
6	0.9112	0.9501	0.9304
8	0.9103	0.9494	0.9297

where t is the time which is necessary for training the ANN by the serial algorithm.

So it is possible to conclude that the usage of parallel algorithms of training ANNs allows to lower capacity and time expenses. So parallel algorithms are very effective.

References

- [1] Горбань А.Н., Дунин-Барковский В.Л., Кирдин А.Н., Миркес Е.М., Новоходько А.Ю., Россиев Д.А., Терехов С.А., Сенашова М.Ю., Царегородцев В.Г. и др. Нейроинформатика // Новосибирск: Наука, 1998, – 296 с. (*Gorban A.N., Dunin-Barkovskiy V.L., Kiridin A.N., Mirkes E.M., Novahodko A.Y., Rossiev D.A., Terehov S.A., Senashova M.Y., Tcaregorodtcev V.G. and etc. Neural computer science // Novosibirsk: Nauka, 1998 – 296 p.*)
- [2] Зенкова Н.А. Моделирование на основе аппарата искусственных нейронных сетей как метод исследования в психологической науке. // Вестн. Тамб. ун-та. Сер. Естеств. и техн. науки. Тамбов: 2009. Т. 14, Вып. 3. – С. 577-591. (*Zenkova, N.n»„. The simulation based on artificial neural networks how discussion method in psychology science. // Tambov University Reports. Series: Natural and Technical Sciences. Tambov, 2009. Vol. 14, Issue. 3 P. – 577-591.*)
- [3] Osowsky S. Sieci neuronowe w ujeciu algorytmicznym // Warszawa. 1996.
- [4] Крючин О.В. Использование кластерных систем для обучения искусственных нейронных сетей при применении параллельного вычисления значения невязки // Наука и образование в развитии промышленной, социальной и экономической сфер регионов России [Электронный ресурс]: II Всероссийские научные Зворыкинские чтения. Сб. тез. докладов II Всероссийской межвузовской научной конференции. -Муром: полиграфический центр МИ ВлГУ, 2010. – 802 с. (*Kryuchin O.V. The usage of computer clusters for the artificial neural networks training using the parallel calculation of inaccuracy // Science and education in the evolution of the industrial, social and economic sphere of Russian regions: II All-Russian interacademic science conference – Murom: polygraphic center MI VlsU, 2010 – 802 p.*)
- [5] Крючин О.В. Параллельные алгоритмы обучения искусственных нейронных сетей // Информационные технологии и математическое моделирование

- (ИТММ-2009) : Материалы VIII Всероссийской научно-практической конференции с международным участием . - Томск: Изд-во Том. Ун-та, 2009. – Ч. 2. С. 241-244. (*Kryuchin O.V. Parallel algorithms of artificial neural networks training // Information technologies and mathematic simulation (ITMS-2009) : Material of VIII All-Russian science-practical conference with international participation – Tomsk: publishing of Tomsk University, 2009 – Part 2. P. 241-244.*)
- [6] Крючин О.В. Разработка параллельных градиентных алгоритмов обучения искусственной нейронной сети // Электронный журнал “Исследовано в России”, 096, С. 1208-1221, 2009 // <http://zhurnal.ape.relarn.ru/articles/2009/096.pdf>. (*Kryuchin O.V. The development of parallel gradient algorithms of the artificial neural networks training // Electronic journal “Investigated in Russia” 096, P. 1208-1221, 2009*) // <http://zhurnal.ape.relarn.ru/articles/2009/096.pdf>
- [7] Солдатова О.П., Семенов В.В. Применение нейронных сетей для решения задач прогнозирования // Электронный журнал “Исследовано в России”, 136, С. 1270-1270, 2009 (*Soldatova O.P., Semenov V.V. The usage of neural networks for forecast problems solving // Electronic journal “Investigated in Russia” 136, P. 1270-1270, 2009*) // <http://zhurnal.ape.relarn.ru/articles/2006/136.pdf>
- [8] Rosenblatt F. Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms. // Washington DC: Spartan Books, 1961
- [9] Rumelhart D.E., Geoffrey E.H., Williams R.J. Learning Internal Representations by Error Propagation. // Parallel distributed processing: Explorations in the microstructure of cognition, Volume 1: Foundations. MIT Press, 1986.
- [10] Cybenko G. Approximation by superpositions of a sigmoidal function Mathematics of Control, Signals, and Systems (MCSS), 2(4), P. 303-314. [Electronic resource] // <http://www.springerlink.com/content/n873j15736072427/>
- [11] Fahlman S.E., Lebiere C. The cascade-correlation learning architecture. Tech. Rep. CMU-CS-90-100, School of Computer Science, Carnegie Mellon University, August 1991.
- [12] Fahlman, S.E. The recurrent cascade-correlation architecture. Tech. Rep. CMU-CS-91-100, School of Computer Science, Carnegie Mellon University, 1991.
- [13] Hoefeld M., Fahlman S.E. Learning with limited numerical precision using the cascade-correlation algorithm. Tech. Rep. CMU-CS-91-130, School of Computer Science, Carnegie Mellon University, 1991.
- [14] Osowsky S., Siwek K. Study of generality ability of neural networks // III Konferencja “Sieci Neuronowe i ich zastosowania”. Kule, 1997

Glossary

- [15] Крючин О.В, Арзамасцев А.А. Прогнозирование котировок валютных пар при помощи искусственной нейронной сети. // Вестн. Тамб. ун-та. Сер. Естеств. и техн. науки. Тамбов: 2009. Т. 14, Вып. 5. С. 591-596. (*Kryuchin O.V., Arzamastsev A.A. The forecasting of currency pairs by artificial neural network. // Tambov University Reports. Series: Natural and Technical Sciences. Tambov, 2009. Vol. 14, Issue. 5. – P. 591-596.*)
- [16] Hertz J., Krogh A., Palmer R. Wstep do teorii obliczen neuronowych. Wyd. II.– Warszawa: WNT, 1995.
- [17] Kryuchin O.V., Arzamastsev A.A., Troitzsch K.G. A universal simulator based on artificial neural networks for computer clusters // Fachbereich Informatik Nr. 2/2011 // [Electronic resource]. http://www.uni-koblenz.de/fb4reports/2011/2011_02_Arbeitsberichte.pdf
- [18] Gill P., Murray W. Wrights M. Practical Optimisation. – N.Y.: Academic Press, 1981.
- [19] Widrow B., Stearns S. Adaptive signal processing. – N.Y.: Prentice Hall, 1985.
- [20] Fahlman S.E. Faster-learning variations on back-propagation: An empirical study. In 1988 Connectionist Models Summer School (San Mateo, CA, 1988), T. J. S. G. E. Hinton and D. S. Touretzky, Eds., Morgan Kaufmann.
- [21] Veith A.C., Holmes G.A. A modified quickprop algorithm // Neural Computation, Vol. 3, 1991. – P. 310-311.
- [22] Fahlman S.E. An empirical study of learning speed in back-propagation networks. Technical report, CMU-CS-88-162, Carnegie-Mellon University, 1988.
- [23] Riedmiller M., Brawn H. RPROP – a fast adaptive learning algorithms. Technacal Report, Karlsruhe: University Karlsruhe, 1992.
- [24] Zadeh L.A., The concept of linguistic variable and its application to approximate reasoning. Part 1-3 // Information Sciences, 1975. – P. 199-249.
- [25] Riedmiller M. Untersuchungen zu konvergenz und generalisierungsverhalten uberwachter lernverfahren mit dem SNNS. In Proceedings of the SNNS. 1993.
- [26] Крючин О.В. Разработка параллельных эвристических алгоритмов подбора весовых коэффициентов искусственной нейронной сети // М.: Информатика и ее применение, Т. 4, Вып. 2, 2010. – С. 53-56. (*Kryuchin O.V. The development of parallel heuristic algorithms of selection of weight coefficients of the artificial neural network // Moscow : Computer science and its adaptation, V. 4, Issue 2, 2010. – P. 53-56.*)

Glossary

Glossary

$C_{\varepsilon k}^{(\iota)}$	the number of operations which are executed in the ι step of parallel algorithm of the calculation of the inaccuracy in the k -th processor, 12
$C_{wFk}^{(\iota)}$	the number of operations which are executed in the ι step of parallel full enumeration algorithm in the k -th processor, 16
$C_{wGk}^{(\iota)}$	the number of operations which are executed in the ι step of parallel gradient algorithm in the k -th processor, 21
F	the function which calculates output values of the ANN, 10, 11
I_F	the number of iterations which are done by full enumeration algorithm, 15
I_G	is the number of iterations which are done by the gradient algorithm, 21
I_ε	the number of calculation of the inaccuracy in the parallel algorithm of the calculation of the target function value, 14
J_F	the number of iterations which are done by parallel full enumeration algorithm in the non-lead processor, 15
L	the number of neurons in the input layer, 2
M	the number of pattern rows which are used by the non-lead processor for the calculation of the inaccuracy in the parallel calculation of the target function value, 11
N	the number of rows in a pattern, 10
N_L	the number of layers in the network, 2
N_x	the number of neuron input, 7
P	the number of neurons in the output layer, 2
Z	the number of operations which are executed by the parallel algorithm (additive operation are recuted to multiplicative operations), 10
Z_ε	the number of operations which are necessary for the parallel calculation of the calculation of the target function value (additive operation are recuted to multiplicative operations), 11
Z_{wF}	the number of operations which are executed by the parallel full enumeration algorithm (additive operation are recuted to multiplicative operations), 17

Glossary

Z_{wG}	the number of operations which are executed by the parallel gradient algorithm (additive operation are recuted to multiplicative operations), 21
Δw_i	the value of the change of the i -th weight coefficient, 18
$\alpha(Z)$	the coefficient of the efficiency of the parallel algorithm (additive operation are recuted to multiplicative operations), 10
$\alpha_\varepsilon(Z)$	the coefficient of the efficiency of the parallel algorithm of the calculation of the target function value, 14
$\alpha_{wF}(Z)$	the coefficient of the efficiency of the parallel full enumeration algorithm (additive operation are recuted to multiplicative operations), 17
$\alpha_{wGQ}(Z)$	the coefficient of the efficiency of the parallel gradient algorithm of QuickProp and the steepest descent (additive operation are recuted to multiplicative operations), 21
$\alpha_{wGR}(Z)$	the coefficient of the efficiency of the parallel gradient algorithm of RPROP (additive operation are recuted to multiplicative operations), 21
γ	the function which returns the number of operations which are necessary for the sending data from one processor to other, 12
$\hat{C}_{\varepsilon 0}$	the number of operations which are executed by the lead processor for the sending weight coefficients to all other processors and calculates inaccuracy ε_0 by the algorithm which parallel calculates the target function value, 14
$\hat{C}_{\varepsilon k}$	the number of operations which are executed by non-lead processors for the receiving the vector of weight coefficients, the calculating the value of the inaccuracy ε_k and the sending result to the lead processor by the algorithm which parallel calculates the target function value, 14
\hat{C}_{wF0}	the number of operations which are executed by the lead processor in first three steps $(C_{wF0})^{(1)} + C_{wF0}^{(2)} + C_{wF0}^{(3)}$ by the parallel full enumeration algorithm, 17

Glossary

\hat{C}_{wFk}	the number of operations which are executed by non-lead processors in first four steps $(C_{wFk})^{(1)} + C_{wFk})^{(2)} + C_{wFk})^{(3)} + C_{wFk})^{(4)}$ by the parallel full enumeration algorithm, 17
\hat{C}_{wG}	the number of operations which are waited by the lead processor before the beginning receiving weight coefficients in parallel gradient algorithms, 21
\hat{J}_F	the number of iterations which are done by parallel full enumeration algorithm in the lead processor, 15
\hat{M}	the number of pattern rows which are used by the lead processor for the calculation of the inaccuracy in the parallel calculation of the target function value, 11
\hat{N}_i	the number of neurons in the i -th layer, 2
\hat{N}_w	the number of weight coefficients (and elements of the gradient vector) which are calculated in the lead processor by the parallel gradient algorithm, 20
\hat{P}	the number of pattern elements which are sent by the lead processor to each nonzero process in the parallel calculation of the target function value, 12
$\hat{\zeta}_\varepsilon$	the number of additive operations which are necessary for the calculation of the target function value, 11
$\hat{\zeta}_n$	the number of additive operations which are necessary for the calculation of the output values of a neuron, 7
$\hat{\zeta}_n(\mu_i)$	the number of additive operations which are necessary for the calculation of the output values of the i -th neuron, 9
$\hat{\zeta}_y$	the number of additive operations which are necessary for the calculation of the output values of an ANN, 9
$\hat{\zeta}_{yC}$	the number of additive operations which are necessary for the calculation of the output values of a cascade-correlation network, 9
$\hat{\zeta}_{yM}$	the number of additive operations which are necessary for the calculation of the output values of a multilayer perceptron, 9

Glossary

$\hat{\zeta}_{yV}$	the number of additive operations which are necessary for the calculation of the output values of a Volterra network, 9
\hat{l}_w	the number of weight coefficients (and elements of the gradient vector) which are calculated in the non-lead processor by the parallel gradient algorithm, 20
λ_G	is the number of operations which are done by the gradient algorithm and which are not operations of the parallel calculation of weight coefficients, 21
λ_ε	the number of operations of the parallel algorithms (of the calculation of the target function value) which are not belong to the calculation of the inaccuracy, 14
σ	the coefficient of the reduction of additive operations to multiplicative operations, 10
$\vec{\mu}$	the activation functions vector, 9
$\vec{\nabla}_\varepsilon$	the vector of the gradient, 18
\vec{p}	the vector of functions of the changing weight coefficients, 18
\vec{s}	the vector of the training, 18
\vec{w}	the vector of weight coefficients, 5
\vec{x}	the vector of input values, 2
\vec{y}	the vector of output values, 2
ζ_ε	the number of multiplicative operations which are necessary for the calculation of the target function value, 11
ζ_n	the number of multiplicative operations which are necessary for the calculation of the output values of a neuron, 7
$\zeta_n(\mu_i)$	the number of multiplicative operations which are necessary for the calculation of the output values of the i -th neuron, 9
ζ_y	the number of multiplicative operations which are necessary for the calculation of the output values of an ANN, 9
ζ_{yC}	the number of multiplicative operations which are necessary for the calculation of the output values of a cascade-correlation network, 9
ζ_{yM}	the number of multiplicative operations which are necessary for the calculation of the output values of a multilayer perceptron, 9

Glossary

ζ_{yV}	the number of multiplicative operations which are necessary for the calculation of the output values of a Volterra network, 9
c_f	the coefficient of an activation function, 7
d	an output pattern matrix, 10
l_i^{down}	the lower limits of the i -th weight coefficient, 15
l_i^{up}	the upper limits of the i -th weight coefficient, 15
l_μ	the total number of neurons in a network, 9
l_w	the number of weights coefficients, 12
n	the number of used processors, 11
p_E	the external impulse, 7
p_I	the internal impulse, 7
q	the coefficient of the moment, 18
t	the points of time, 5
v	the rate of an interconnect, 12
z	the number of operations which are executed by the serial algorithm (additive operation are recuted to multiplicative operations), 10
z_ε	the number of operations which are necessary for the calculation of the serial calculation of the target function value (additive operation are recuted to multiplicative operations), 11
z_n	the number of operations which are necessary for the calculation of the output values of a neuron (additive operation are reduced to multiplicative operations), 7
$z_n(\mu_i)$	the number of operations which are necessary for the calculation of the output values of the i -th neuron (additive operation are recuted to multiplicative operations), 9
z_y	the number of operations which are necessary for the calculation of the output values of an ANN (additive operation are recuted to multiplicative operations), 9
z_{wF}	the number of operations which are executed by the serial full enumeration algorithm (additive operation are recuted to multiplicative operations), 17
z_{wG}	the number of operations which are executed by the serial gradient algorithm (additive operation are recuted to multiplicative operations), 21

z_{yC}	the number of operations which are necessary for the calculation of the output values of a cascade-correlation network (additive operation are recuted to multiplicative operations), 9
z_{yM}	the number of operations which are necessary for the calculation of the output values of a multilayer perceptron (additive operation are recuted to multiplicative operations), 9
z_{yV}	the number of operations which are necessary for the calculation of the output values of a Volterra network (additive operation are recuted to multiplicative operations), 9

Bisher erschienen

Arbeitsberichte aus dem Fachbereich Informatik

(<http://www.uni-koblenz-landau.de/koblenz/fb4/publications/Reports/arbeitsberichte>)

Oleg V. Kryuchin, Alexander A. Arzamastsev, Klaus G. Troitzsch, Comparing the efficiency of serial and parallel algorithms for training artificial neural networks using computer clusters, Arbeitsberichte aus dem Fachbereich Informatik, 13/2011

Oleg V. Kryuchin, Alexander A. Arzamastsev, Klaus G. Troitzsch, A parallel algorithm for selecting activation functions of an artificial network, Arbeitsberichte aus dem Fachbereich Informatik 12/2011

Katharina Bräunlich, Rüdiger Grimm, Andreas Kasten, Sven Vowé, Nico Jahn, Der neue Personalausweis zur Authentifizierung von Wählern bei Onlinewahlen, Arbeitsberichte aus dem Fachbereich Informatik 11/2011

Daniel Eißing, Ansgar Scherp, Steffen Staab, Formal Integration of Individual Knowledge Work and Organizational Knowledge Work with the Core Ontology *strukt*, Arbeitsberichte aus dem Fachbereich Informatik 10/2011

Bernhard Reinert, Martin Schumann, Stefan Müller, Combined Non-Linear Pose Estimation from Points and Lines, Arbeitsberichte aus dem Fachbereich Informatik 9/2011

Tina Walber, Ansgar Scherp, Steffen Staab, Towards the Understanding of Image Semantics by Gaze-based Tag-to-Region Assignments, Arbeitsberichte aus dem Fachbereich Informatik 8/2011

Alexander Kleinen, Ansgar Scherp, Steffen Staab, Mobile Facets – Faceted Search and Exploration of Open Social Media Data on a Touchscreen Mobile Phone, Arbeitsberichte aus dem Fachbereich Informatik 7/2011

Anna Lantsberg, Klaus G. Troitzsch, Towards A Methodology of Developing Models of E-Service Quality Assessment in Healthcare, Arbeitsberichte aus dem Fachbereich Informatik 6/2011

Ansgar Scherp, Carsten Saathoff, Thomas Franz, Steffen Staab, Designing Core Ontologies, Arbeitsberichte aus dem Fachbereich Informatik 5/2011

Oleg V. Kryuchin, Alexander A. Arzamastsev, Klaus G. Troitzsch, The prediction of currency exchange rates using artificial neural networks, Arbeitsberichte aus dem Fachbereich Informatik 4/2011

Klaus G. Troitzsch, Anna Lantsberg, Requirements for Health Care Related Websites in Russia: Results from an Analysis of American, British and German Examples, Arbeitsberichte aus dem Fachbereich Informatik 3/2011

Klaus G. Troitzsch, Oleg Kryuchin, Alexander Arzamastsev, A universal simulator based on artificial neural networks for computer clusters, Arbeitsberichte aus dem Fachbereich Informatik 2/2011

Klaus G. Troitzsch, Natalia Zenkova, Alexander Arzamastsev, Development of a technology of designing intelligent information systems for the estimation of social objects, Arbeitsberichte aus dem Fachbereich Informatik 1/2011

Kurt Lautenbach, A Petri Net Approach for Propagating Probabilities and Mass Functions, Arbeitsberichte aus dem Fachbereich Informatik 13/2010

Claudia Schon, Linkless Normal Form for ALC Concepts, Arbeitsberichte aus dem Fachbereich Informatik 12/2010

Alexander Hug, Informatik hautnah erleben, Arbeitsberichte aus dem Fachbereich Informatik 11/2010

Marc Santos, Harald F.O. von Kortzfleisch, Shared Annotation Model – Ein Datenmodell für kollaborative Annotationen, Arbeitsberichte aus dem Fachbereich Informatik 10/2010

Gerd Gröner, Steffen Staab, Categorization and Recognition of Ontology Refactoring Pattern, Arbeitsberichte aus dem Fachbereich Informatik 9/2010

Daniel Eißing, Ansgar Scherp, Carsten Saathoff, Integration of Existing Multimedia Metadata Formats and Metadata Standards in the M3O, Arbeitsberichte aus dem Fachbereich Informatik 8/2010

Stefan Scheglmann, Ansgar Scherp, Steffen Staab, Model-driven Generation of APIs for OWL-based Ontologies, Arbeitsberichte aus dem Fachbereich Informatik 7/2010

Daniel Schmeiß, Ansgar Scherp, Steffen Staab, Integrated Mobile Visualization and Interaction of Events and POIs, Arbeitsberichte aus dem Fachbereich Informatik 6/2010

Rüdiger Grimm, Daniel Pähler, E-Mail-Forensik – IP-Adressen und ihre Zuordnung zu Internet-Teilnehmern und ihren Standorten, Arbeitsberichte aus dem Fachbereich Informatik 5/2010

Christoph Ringelstein, Steffen Staab, PAPEL: Syntax and Semantics for Provenance-Aware Policy Definition, Arbeitsberichte aus dem Fachbereich Informatik 4/2010

Nadine Lindermann, Sylvia Valcárcel, Harald F.O. von Kortzfleisch, Ein Stufenmodell für kollaborative offene Innovationsprozesse in Netzwerken kleiner und mittlerer Unternehmen mit Web 2.0, Arbeitsberichte aus dem Fachbereich Informatik 3/2010

Maria Wimmer, Dagmar Lück-Schneider, Uwe Brinkhoff, Erich Schweighofer, Siegfried Kaiser, Andreas Wieber, Fachtagung Verwaltungsinformatik FTVI Fachtagung Rechtsinformatik FTRI 2010, Arbeitsberichte aus dem Fachbereich Informatik 2/2010

Max Braun, Ansgar Scherp, Steffen Staab, Collaborative Creation of Semantic Points of Interest as Linked Data on the Mobile Phone, Arbeitsberichte aus dem Fachbereich Informatik 1/2010

Marc Santos, Einsatz von „Shared In-situ Problem Solving“ Annotationen in kollaborativen Lern- und Arbeitsszenarien, Arbeitsberichte aus dem Fachbereich Informatik 20/2009

Carsten Saathoff, Ansgar Scherp, Unlocking the Semantics of Multimedia Presentations in the Web with the Multimedia Metadata Ontology, Arbeitsberichte aus dem Fachbereich Informatik 19/2009

Christoph Kahle, Mario Schaarschmidt, Harald F.O. von Kortzfleisch, Open Innovation: Kundenintegration am Beispiel von IPTV, Arbeitsberichte aus dem Fachbereich Informatik 18/2009

Dietrich Paulus, Lutz Prieße, Peter Decker, Frank Schmitt, Pose-Tracking Forschungsbericht, Arbeitsberichte aus dem Fachbereich Informatik 17/2009

Andreas Fuhr, Tassilo Horn, Andreas Winter, Model-Driven Software Migration Extending SOMA, Arbeitsberichte aus dem Fachbereich Informatik 16/2009

Eckhard Großmann, Sascha Strauß, Tassilo Horn, Volker Riediger, Abbildung von grUML nach XSD soamig, Arbeitsberichte aus dem Fachbereich Informatik 15/2009

Kerstin Falkowski, Jürgen Ebert, The STOR Component System Interim Report, Arbeitsberichte aus dem Fachbereich Informatik 14/2009

Sebastian Magnus, Markus Maron, An Empirical Study to Evaluate the Location of Advertisement Panels by Using a Mobile Marketing Tool, Arbeitsberichte aus dem Fachbereich Informatik 13/2009

Sebastian Magnus, Markus Maron, Konzept einer Public Key Infrastruktur in iCity, Arbeitsberichte aus dem Fachbereich Informatik 12/2009

Sebastian Magnus, Markus Maron, A Public Key Infrastructure in Ambient Information and Transaction Systems, Arbeitsberichte aus dem Fachbereich Informatik 11/2009

Ammar Mohammed, Ulrich Furbach, Multi-agent systems: Modeling and Virification using Hybrid Automata, Arbeitsberichte aus dem Fachbereich Informatik 10/2009

Andreas Sprotte, Performance Measurement auf der Basis von Kennzahlen aus betrieblichen Anwendungssystemen: Entwurf eines kennzahlengestützten Informationssystems für einen Logistikdienstleister, Arbeitsberichte aus dem Fachbereich Informatik 9/2009

Gwendolin Garbe, Tobias Hausen, Process Commodities: Entwicklung eines Reifegradmodells als Basis für Outsourcingentscheidungen, Arbeitsberichte aus dem Fachbereich Informatik 8/2009

Petra Schubert et. al., Open-Source-Software für das Enterprise Resource Planning, Arbeitsberichte aus dem Fachbereich Informatik 7/2009

Ammar Mohammed, Frieder Stolzenburg, Using Constraint Logic Programming for Modeling and Verifying Hierarchical Hybrid Automata, Arbeitsberichte aus dem Fachbereich Informatik 6/2009

Tobias Kippert, Anastasia Meletiadiou, Rüdiger Grimm, Entwurf eines Common Criteria-Schutzprofils für Router zur Abwehr von Online-Überwachung, Arbeitsberichte aus dem Fachbereich Informatik 5/2009

Hannes Schwarz, Jürgen Ebert, Andreas Winter, Graph-based Traceability – A Comprehensive Approach. Arbeitsberichte aus dem Fachbereich Informatik 4/2009

Anastasia Meletiadiou, Simone Müller, Rüdiger Grimm, Anforderungsanalyse für Risk-Management-Informationssysteme (RMIS), Arbeitsberichte aus dem Fachbereich Informatik 3/2009

Ansgar Scherp, Thomas Franz, Carsten Saathoff, Steffen Staab, A Model of Events based on a Foundational Ontology, Arbeitsberichte aus dem Fachbereich Informatik 2/2009

Frank Bohdanovicz, Harald Dickel, Christoph Steigner, Avoidance of Routing Loops, Arbeitsberichte aus dem Fachbereich Informatik 1/2009

Stefan Ameling, Stephan Wirth, Dietrich Paulus, Methods for Polyp Detection in Colonoscopy Videos: A Review, Arbeitsberichte aus dem Fachbereich Informatik 14/2008

Tassilo Horn, Jürgen Ebert, Ein Referenzschema für die Sprachen der IEC 61131-3, Arbeitsberichte aus dem Fachbereich Informatik 13/2008

Thomas Franz, Ansgar Scherp, Steffen Staab, Does a Semantic Web Facilitate Your Daily Tasks?, Arbeitsberichte aus dem Fachbereich Informatik 12/2008

Norbert Frick, Künftige Anfordeungen an ERP-Systeme: Deutsche Anbieter im Fokus, Arbeitsberichte aus dem Fachbereich Informatik 11/2008

Jürgen Ebert, Rüdiger Grimm, Alexander Hug, Lehramtsbezogene Bachelor- und Masterstudiengänge im Fach Informatik an der Universität Koblenz-Landau, Campus Koblenz, Arbeitsberichte aus dem Fachbereich Informatik 10/2008

Mario Schaarschmidt, Harald von Kortzfleisch, Social Networking Platforms as Creativity Fostering Systems: Research Model and Exploratory Study, Arbeitsberichte aus dem Fachbereich Informatik 9/2008

Bernhard Schueler, Sergej Sizov, Steffen Staab, Querying for Meta Knowledge, Arbeitsberichte aus dem Fachbereich Informatik 8/2008

Stefan Stein, Entwicklung einer Architektur für komplexe kontextbezogene Dienste im mobilen Umfeld, Arbeitsberichte aus dem Fachbereich Informatik 7/2008

Matthias Bohnen, Lina Brühl, Sebastian Bzdak, RoboCup 2008 Mixed Reality League Team Description, Arbeitsberichte aus dem Fachbereich Informatik 6/2008

Bernhard Beckert, Reiner Hähnle, Tests and Proofs: Papers Presented at the Second International Conference, TAP 2008, Prato, Italy, April 2008, Arbeitsberichte aus dem Fachbereich Informatik 5/2008

Klaas Dellschaft, Steffen Staab, Unterstützung und Dokumentation kollaborativer Entwurfs- und Entscheidungsprozesse, Arbeitsberichte aus dem Fachbereich Informatik 4/2008

Rüdiger Grimm: IT-Sicherheitsmodelle, Arbeitsberichte aus dem Fachbereich Informatik 3/2008

Rüdiger Grimm, Helge Hundacker, Anastasia Meletiadou: Anwendungsbeispiele für Kryptographie, Arbeitsberichte aus dem Fachbereich Informatik 2/2008

Markus Maron, Kevin Read, Michael Schulze: CAMPUS NEWS – Artificial Intelligence Methods Combined for an Intelligent Information Network, Arbeitsberichte aus dem Fachbereich Informatik 1/2008

Lutz Priebe, Frank Schmitt, Patrick Sturm, Haojun Wang: BMBF-Verbundprojekt 3D-RETISEG Abschlussbericht des Labors Bilderkennen der Universität Koblenz-Landau, Arbeitsberichte aus dem Fachbereich Informatik 26/2007

Stephan Philippi, Alexander Pinl: Proceedings 14. Workshop 20.-21. September 2007 Algorithmen und Werkzeuge für Petrinetze, Arbeitsberichte aus dem Fachbereich Informatik 25/2007

Ulrich Furbach, Markus Maron, Kevin Read: CAMPUS NEWS – an Intelligent Bluetooth-based Mobile Information Network, Arbeitsberichte aus dem Fachbereich Informatik 24/2007

Ulrich Furbach, Markus Maron, Kevin Read: CAMPUS NEWS - an Information Network for Pervasive Universities, Arbeitsberichte aus dem Fachbereich Informatik 23/2007

Lutz Priebe: Finite Automata on Unranked and Unordered DAGs Extended Version, Arbeitsberichte aus dem Fachbereich Informatik 22/2007

Mario Schaarschmidt, Harald F.O. von Kortzfleisch: Modularität als alternative Technologie- und Innovationsstrategie, Arbeitsberichte aus dem Fachbereich Informatik 21/2007

Kurt Lautenbach, Alexander Pinl: Probability Propagation Nets, Arbeitsberichte aus dem Fachbereich Informatik 20/2007

Rüdiger Grimm, Farid Mehr, Anastasia Meletiadou, Daniel Pähler, Ilka Uerz: SOA-Security, Arbeitsberichte aus dem Fachbereich Informatik 19/2007

Christoph Wernhard: Tableaux Between Proving, Projection and Compilation, Arbeitsberichte aus dem Fachbereich Informatik 18/2007

Ulrich Furbach, Claudia Obermaier: Knowledge Compilation for Description Logics, Arbeitsberichte aus dem Fachbereich Informatik 17/2007

Fernando Silva Parreiras, Steffen Staab, Andreas Winter: TwoUse: Integrating UML Models and OWL Ontologies, Arbeitsberichte aus dem Fachbereich Informatik 16/2007

Rüdiger Grimm, Anastasia Meletiadou: Rollenbasierte Zugriffskontrolle (RBAC) im Gesundheitswesen, Arbeitsberichte aus dem Fachbereich Informatik 15/2007

Ulrich Furbach, Jan Murray, Falk Schmidsberger, Frieder Stolzenburg: Hybrid Multiagent Systems with Timed Synchronization-Specification and Model Checking, Arbeitsberichte aus dem Fachbereich Informatik 14/2007

Björn Pelzer, Christoph Wernhard: System Description: "E-KRHyper", Arbeitsberichte aus dem Fachbereich Informatik, 13/2007

Ulrich Furbach, Peter Baumgartner, Björn Pelzer: Hyper Tableaux with Equality, Arbeitsberichte aus dem Fachbereich Informatik, 12/2007

Ulrich Furbach, Markus Maron, Kevin Read: Location based Information systems, Arbeitsberichte aus dem Fachbereich Informatik, 11/2007

Philipp Schaer, Marco Thum: State-of-the-Art: Interaktion in erweiterten Realitäten, Arbeitsberichte aus dem Fachbereich Informatik, 10/2007

Ulrich Furbach, Claudia Obermaier: Applications of Automated Reasoning, Arbeitsberichte aus dem Fachbereich Informatik, 9/2007

Jürgen Ebert, Kerstin Falkowski: A First Proposal for an Overall Structure of an Enhanced Reality Framework, Arbeitsberichte aus dem Fachbereich Informatik, 8/2007

Lutz Priese, Frank Schmitt, Paul Lemke: Automatische See-Through Kalibrierung, Arbeitsberichte aus dem Fachbereich Informatik, 7/2007

Rüdiger Grimm, Robert Krimmer, Nils Meißner, Kai Reinhard, Melanie Volkamer, Marcel Weinand, Jörg Helbach: Security Requirements for Non-political Internet Voting, Arbeitsberichte aus dem Fachbereich Informatik, 6/2007

Daniel Bildhauer, Volker Riediger, Hannes Schwarz, Sascha Strauß, „grUML – Eine UML-basierte Modellierungssprache für T-Graphen“, Arbeitsberichte aus dem Fachbereich Informatik, 5/2007

Richard Arndt, Steffen Staab, Raphaël Troncy, Lynda Hardman: Adding Formal Semantics to MPEG-7: Designing a Well Founded Multimedia Ontology for the Web, Arbeitsberichte aus dem Fachbereich Informatik, 4/2007

Simon Schenk, Steffen Staab: Networked RDF Graphs, Arbeitsberichte aus dem Fachbereich Informatik, 3/2007

Rüdiger Grimm, Helge Hundacker, Anastasia Meletiadou: Anwendungsbeispiele für Kryptographie, Arbeitsberichte aus dem Fachbereich Informatik, 2/2007

Anastasia Meletiadou, J. Felix Hampe: Begriffsbestimmung und erwartete Trends im IT-Risk-Management, Arbeitsberichte aus dem Fachbereich Informatik, 1/2007

„Gelbe Reihe“

(<http://www.uni-koblenz.de/fb4/publikationen/gelbereihe>)

Lutz Priese: Some Examples of Semi-rational and Non-semi-rational DAG Languages. Extended Version, Fachberichte Informatik 3-2006

Kurt Lautenbach, Stephan Philippi, and Alexander Pinl: Bayesian Networks and Petri Nets, Fachberichte Informatik 2-2006

Rainer Gimnich and Andreas Winter: Workshop Software-Reengineering und Services, Fachberichte Informatik 1-2006

Kurt Lautenbach and Alexander Pinl: Probability Propagation in Petri Nets, Fachberichte Informatik 16-2005

Rainer Gimnich, Uwe Kaiser, and Andreas Winter: 2. Workshop "Reengineering Prozesse" – Software Migration, Fachberichte Informatik 15-2005

Jan Murray, Frieder Stolzenburg, and Toshiaki Arai: Hybrid State Machines with Timed Synchronization for Multi-Robot System Specification, Fachberichte Informatik 14-2005

Reinhold Letz: FTP 2005 – Fifth International Workshop on First-Order Theorem Proving, Fachberichte Informatik 13-2005

Bernhard Beckert: TABLEAUX 2005 – Position Papers and Tutorial Descriptions, Fachberichte Informatik 12-2005

Dietrich Paulus and Detlev Droege: Mixed-reality as a challenge to image understanding and artificial intelligence, Fachberichte Informatik 11-2005

Jürgen Sauer: 19. Workshop Planen, Scheduling und Konfigurieren / Entwerfen, Fachberichte Informatik 10-2005

Pascal Hitzler, Carsten Lutz, and Gerd Stumme: Foundational Aspects of Ontologies, Fachberichte Informatik 9-2005

Joachim Baumeister and Dietmar Seipel: Knowledge Engineering and Software Engineering, Fachberichte Informatik 8-2005

Benno Stein and Sven Meier zu Eißel: Proceedings of the Second International Workshop on Text-Based Information Retrieval, Fachberichte Informatik 7-2005

Andreas Winter and Jürgen Ebert: Metamodel-driven Service Interoperability, Fachberichte Informatik 6-2005

Joschka Boedecker, Norbert Michael Mayer, Masaki Ogino, Rodrigo da Silva Guerra, Masaaki Kikuchi, and Minoru Asada: Getting closer: How Simulation and Humanoid League can benefit from each other, Fachberichte Informatik 5-2005

Torsten Gipp and Jürgen Ebert: Web Engineering does profit from a Functional Approach, Fachberichte Informatik 4-2005

Oliver Obst, Anita Maas, and Joschka Boedecker: HTN Planning for Flexible Coordination Of Multiagent Team Behavior, Fachberichte Informatik 3-2005

Andreas von Hessling, Thomas Kleemann, and Alex Sinner: Semantic User Profiles and their Applications in a Mobile Environment, Fachberichte Informatik 2-2005

Heni Ben Amor and Achim Rettinger: Intelligent Exploration for Genetic Algorithms – Using Self-Organizing Maps in Evolutionary Computation, Fachberichte Informatik 1-2005