

Parallel Implementation of the Givens Rotations in the Neural Network Learning Algorithm

Jarosław Bilski¹(✉), Bartosz Kowalczyk¹, and Jacek M. Żurada²

¹ Institute of Computational Intelligence,
Częstochowa University of Technology, Częstochowa, Poland
{Jaroslaw.Bilski,Bartosz.Kowalczyk}@iisi.pcz.pl

² Department Electrical and Computer Engineering,
University of Louisville, Louisville, KY 40292, USA
jacek.zurada@louisville.edu

Abstract. The paper describes a parallel feed-forward neural network training algorithm based on the QR decomposition with the use of the Givens rotation. The beginning brings a brief mathematical background on Givens rotation matrices and elimination step. Then the error criterion and its necessary transformations for the QR decomposition are presented. The paper's core holds an essential explanation to accomplish hardware-based parallel implementation. The paper concludes with a theoretical description of speed improvement gained by parallel implementation of the Givens reduction in the QR decomposition process.

Keywords: Feed-forward neural network · Neural network training algorithm · Optimization · QR decomposition · Givens rotation · Parallel Givens rotation · Parallel Givens flow

1 Introduction

Artificial neural networks are finding countless applications in both, the worlds of science and industry. This generates an everlasting demand for scientific research in those areas as in [2,3,15–17,19,23,24]. One of the most significant branches is the neural network teaching process. For that purpose many algorithms have been developed [7,10,22]. Some of them are broadly known, easy to implement but require a significant amount of time to establish convergence [12,21]. Other algorithms are much faster at the cost of more complex implementation and utilization of many more resources [8,14]. The next important branch in the area of artificial neural network research is optimization of already-known teaching algorithms. A well designed practice for that purpose is parallelization of the serial algorithm. Many research projects in this area have been attempted as in [4,6,9,11]. This paper focuses on delivering a hardware-based parallelization concept of feed-forward neural network teaching algorithm using the QR decomposition based on the Givens rotation. The principal goal of this idea is to complete the QR decomposition process in a lower iteration count than required by the serial variant of this algorithm.

2 Givens Elimination Step

The Givens reduction is the process of applying orthogonal matrix \mathbf{G}_{pq} (shown in Eq. 1) by left-sided multiplication of rotated vector \mathbf{a} .

$$\mathbf{G}_{pq} = \begin{bmatrix} 1 & & \cdots & & 0 \\ & \ddots & & & \\ & & c & \cdots & s \\ \vdots & & \vdots & \ddots & \vdots \\ & & -s & \cdots & c \\ 0 & & & \cdots & 1 \end{bmatrix} \begin{matrix} p \\ q \end{matrix} \quad (1)$$

$$\mathbf{a} \rightarrow \bar{\mathbf{a}} = \mathbf{G}_{pq}\mathbf{a}. \quad (2)$$

Equation 2 performs the following operations

$$\begin{aligned} \bar{a}_p &= ca_p + sa_q \\ \bar{a}_q &= -sa_p + ca_q \\ \bar{a}_i &= a_i \quad (i \neq p, q; i = 1, \dots, n). \end{aligned} \quad (3)$$

In order to eliminate the value of a_q , parameters c and s should be calculated according to the following equations

$$c = \frac{a_p}{\rho} \quad s = \frac{a_q}{\rho}, \quad (4)$$

where ρ due to numerical error minimization takes the form

$$\rho = \begin{cases} a_p \sqrt{1 + (a_q/a_p)^2}, & \text{for } |a_p| \geq |a_q| \\ a_q \sqrt{1 + (a_p/a_q)^2}, & \text{for } |a_p| < |a_q| \end{cases} \quad (5)$$

When Eqs. 4 and 5 are applied to 3, we obtain the following

$$\begin{aligned} \bar{a}_p &= ca_p + sa_q = \rho \\ \bar{a}_q &= -sa_p + ca_q = 0. \end{aligned} \quad (6)$$

3 Givens QR Decomposition

The QR decomposition is the process of transforming any non-singular matrix into the product of orthogonal \mathbf{Q} and upper-triangle \mathbf{R} matrices

$$\mathbf{A} = \mathbf{Q}\mathbf{R}, \quad (7)$$

where

$$\mathbf{Q}^T \mathbf{Q} = \mathbf{I}, \quad (8)$$

$$\mathbf{Q}^T = \mathbf{Q}^{-1}, \quad (9)$$

$$r_{ij} = 0 \quad \text{for } i > j. \quad (10)$$

The orthogonal matrix \mathbf{Q} does not need to be explicitly determined. It is enough to calculate c and s parameters of their respective rotations and apply them to input matrix \mathbf{A} . Acquisition of all c and s parameters for all \mathbf{G}_{pq} matrices concludes the QR decomposition by transforming matrix \mathbf{A} to an upper-triangle form

$$\mathbf{R} = \mathbf{G}_{m-1} \dots \mathbf{G}_1 \mathbf{A}_1 = \mathbf{G}_{m-1,m} \dots \mathbf{G}_{23} \dots \mathbf{G}_{2m} \mathbf{G}_{12} \dots \mathbf{G}_{1m} \mathbf{A}_1 = \mathbf{Q}^T \mathbf{A} \quad (11)$$

Since there is no need to explicitly create \mathbf{G}_{pq} matrices, the orthogonal matrix \mathbf{Q} can be calculated back from the respective rotations

$$\mathbf{Q} = \mathbf{G}_1^T \dots \mathbf{G}_{m-1}^T = \mathbf{G}_{1m}^T \dots \mathbf{G}_{12}^T \mathbf{G}_{2m}^T \dots \mathbf{G}_{23}^T \dots \mathbf{G}_{m-1,m}^T \quad (12)$$

4 QR Decomposition in Neural Network Weights Update

In order to calculate weight update the following error criterion is given

$$J(n) = \sum_{t=1}^n \lambda^{n-t} \sum_{j=1}^{N_L} \varepsilon_j^{(L)2}(t) = \sum_{t=1}^n \lambda^{n-t} \sum_{j=1}^{N_L} \left[d_j^{(L)}(t) - f(\mathbf{x}^{(L)T}(t) \mathbf{w}_j^{(L)}(n)) \right]^2 \quad (13)$$

By performing the derivation and linearization of Eq. 13 the normal equation is obtained

$$\sum_{t=1}^n \lambda^{n-t} f'^2(s_i^{(l)}(t)) \left[b_i^{(l)}(t) - \mathbf{x}^{(l)T}(t) \mathbf{w}_i^{(l)}(n) \right] \mathbf{x}^{(l)T}(t) = \mathbf{0}. \quad (14)$$

Transforming Eq. 14 to the vector form reveals an entry point to the QR decomposition. The following equations are applicable for each layer, where i denotes the neuron index of layer l .

$$\mathbf{A}_i^{(l)}(n) \mathbf{w}_i^{(l)}(n) = \mathbf{h}_i^{(l)}(n), \quad (15)$$

where

$$\mathbf{A}_i^{(l)}(n) = \sum_{t=1}^n \lambda^{n-t} \mathbf{z}_i^{(l)}(t) \mathbf{z}_i^{(l)T}(t), \quad (16)$$

$$\mathbf{h}_i^{(l)}(n) = \sum_{t=1}^n \lambda^{n-t} f'(s_i^{(l)}(t)) b_i^{(l)}(t) \mathbf{z}_i^{(l)}(t), \quad (17)$$

where

$$\mathbf{z}_i^{(l)}(t) = f' \left(s_i^{(l)}(t) \right) \mathbf{x}^{(l)}(t). \quad (18)$$

$$b_i^{(l)}(n) = \begin{cases} b_i^{(L)}(n) = f^{-1} \left(d_i^{(L)}(n) \right) & \text{for } l = L \\ s_i^{(l)}(n) + e_i^{(l)}(n) & \text{for } l = 1 \dots L-1, \end{cases} \quad (19)$$

$$e_i^{(k)}(n) = \sum_{j=1}^{N_{k+1}} f' \left(s_i^{(k)}(n) \right) w_{ji}^{(k+1)}(n) e_j^{(k+1)}(n) \quad \text{for } k = 1 \dots L-1. \quad (20)$$

Equation 15 is solved by the QR decomposition. The first step is to acquire orthogonal matrix \mathbf{Q}^T and perform left-sided multiplication

$$\mathbf{Q}_i^{(l)T}(n) \mathbf{A}_i^{(l)}(n) \mathbf{w}_i^{(l)}(n) = \mathbf{Q}_i^{(l)T}(n) \mathbf{h}_i^{(l)}(n), \quad (21)$$

$$\mathbf{R}_i^{(l)}(n) \mathbf{w}_i^{(l)}(n) = \mathbf{Q}_i^{(l)T}(n) \mathbf{h}_i^{(l)}(n). \quad (22)$$

Finally, matrix \mathbf{A} is transformed to \mathbf{R} , which is upper-triangle. The update of neuron weights is described by the following equations

$$\hat{\mathbf{w}}_i^{(l)}(n) = \mathbf{R}_i^{(l)-1}(n) \mathbf{Q}_i^{(l)T}(n) \mathbf{h}_i^{(l)}(n), \quad (23)$$

$$\mathbf{w}_i^{(l)}(n) = (1 - \eta) \mathbf{w}_i^{(l)}(n-1) + \eta \hat{\mathbf{w}}_i^{(l)}(n). \quad (24)$$

5 Parallel Implementation

The parallel weight update system consists of three subsystems where each one is executed one by one. All the structures presented in this section are related to single neuron only. A complete layer parallel weight update system is of 3D cuboid structure whose size depends on the layer's neuron count. The first subsystem can be found in Fig. 1. It is responsible for formulating autocorrelation matrix \mathbf{A} and vector \mathbf{h} according to Eqs. 16 and 17. Block **S** executes Eq. 18. The definition of blocks **S**, **A1** and **H1** can be found in Fig. 2. Matrix \mathbf{A} and vector \mathbf{h} determined after the first step cannot be changed until the current iteration of the teaching process is finished as they are used in oncoming iterations. The second system realizes the parallel QR decomposition based on Givens rotation according to Eq. 3, keeping in mind Eq. 6. The second system is supposed to calculate matrix \mathbf{R} and vector $\mathbf{Q}^T \mathbf{h}$, which is done by the structure shown in Fig. 3. Blocks **A2** are holding respective copies of matrix \mathbf{A} . Similarly blocks **H2** are holding their respective copies from vector \mathbf{h} . From Eq. 16 we know that matrix \mathbf{A} is a square matrix of size $N_{l-1} + 1 \times N_{l-1} + 1$, where N_{l-1} denotes input layer vector size. To improve readability of oncoming equations the following substitution is performed $n = N_{l-1} + 1$. In most cases the complete QR decomposition cannot be done in a single step (except $N_{l-1} = 2$). An example parallel flow (for $N_{l-1} = 8$) of the QR decomposition based on the Givens rotation is shown in Fig. 5. Each parallel elimination step affects only two rows of

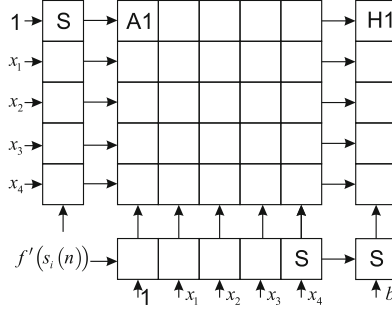


Fig. 1. Structure of the system responsible for formulating matrix \mathbf{A} and vector \mathbf{h} .

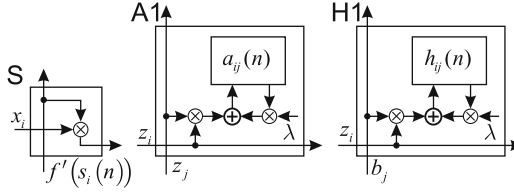


Fig. 2. Definitions of A1, H1 and S processing elements.

matrix \mathbf{A} starting from column k , where $k \in \langle 1, 2, \dots, N_{l-1} - 1 \rangle$. The starting point of a single rotation is represented by objects A2pk and A2qk shown in Fig. 4. Block A2pk calculates length (ρ) of vector $[a_{pk}, a_{qk}]^T$. Then, the value of A2pk block ($a_{pk}(n)$) is replaced by obtained ρ . Simultaneously, value $a_{qk}(n)$ in block A2qk is replaced by 0 according to Eq. 6. Block A2pk is also responsible for determining rotation c and s parameters which are applied for all right-hand objects starting from A2pk and A2qk blocks in matrix \mathbf{A} . Also values p and q of vector \mathbf{h} are directly affected by c and s parameters as shown in Fig. 4. As result of solving the QR decomposition in the second step matrix \mathbf{R} and vector $\mathbf{Q}^T \mathbf{h}$ are obtained. The third-last system is responsible for updating weights of the respective neuron according to Eqs. 23 and 24. The parallel structure that is capable of doing that is presented in Fig. 6. Objects A3a and A3b determine the value of \hat{w}_{ij} according to Eq. 25 while block W performs the weight update. The third system block definitions are shown in Fig. 7.

$$\hat{w}_{mj} = \frac{h_{mj} - \sum_{k=i+1}^n \hat{w}_{ik} a_{ik}}{a_{ii}} \quad (25)$$

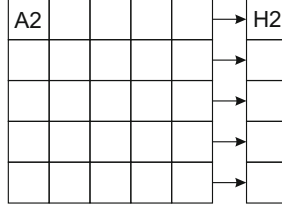


Fig. 3. Structure of the Givens rotation system.

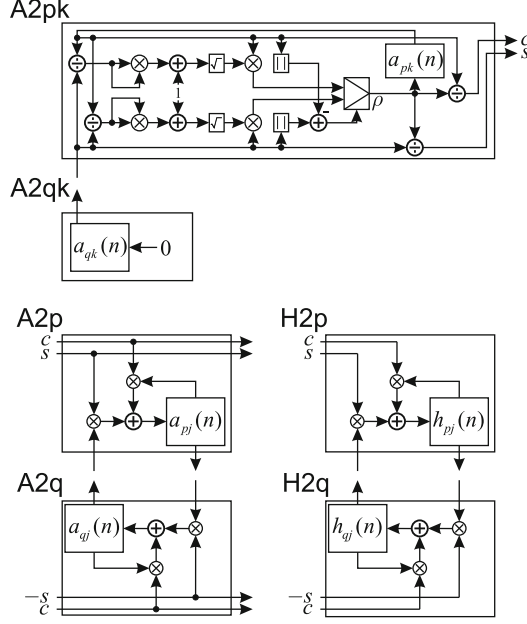


Fig. 4. Blocks of A2 and H2 structures.

6 Simulation Results

Using a parallel variant of the Givens QR decomposition it is possible to achieve the process performance given by Eq. 26

$$\xi_p \leq 2n - b, \quad (26)$$

where $n = N_{l-1} + 1$ and b is given in Table 1. Performance of serial variant is given by Eq. 27

$$\xi_s = \sum_{k=1}^{n-1} n - k. \quad (27)$$

The whole teaching process complexity of serial and parallel variants for single layer l which consists of N_l neurons is shown in Table 2. When a parallel structure

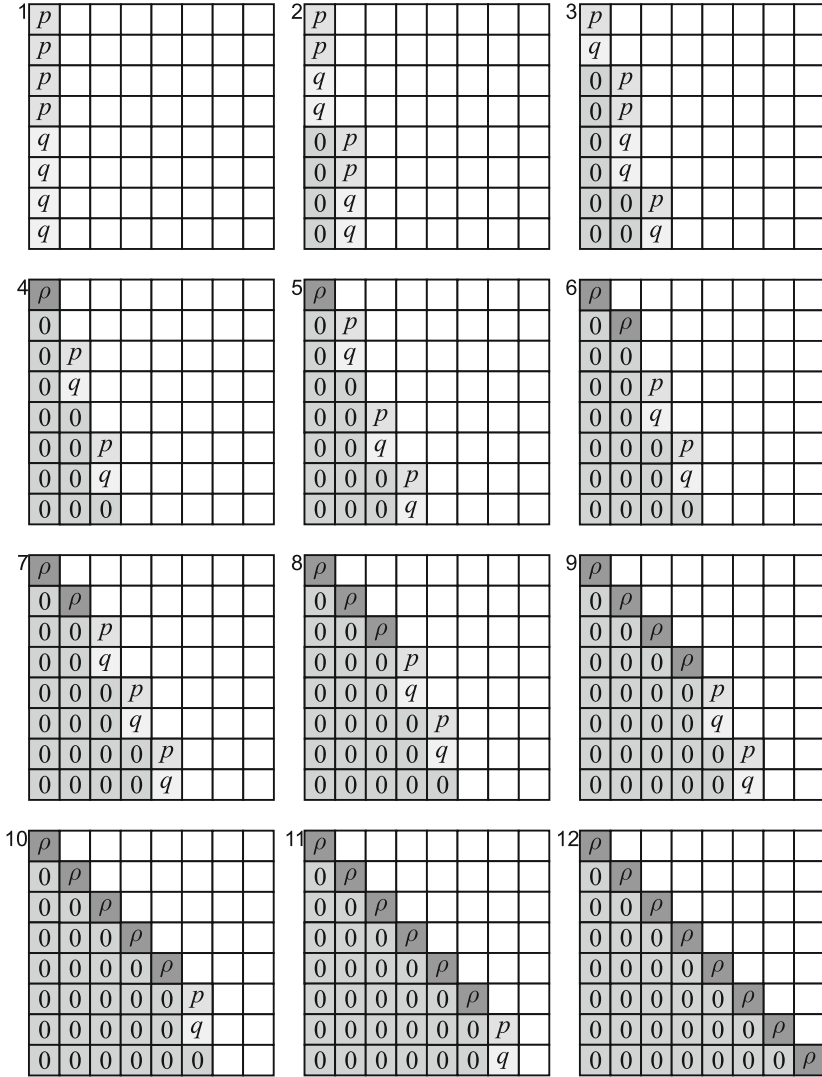


Fig. 5. The parallel QR decomposition based on the Givens rotation flow for 8×8 matrix. The first step is presented in the upper left corner of the figure. Labels A2 and column index k have been omitted to improve picture readability. Each block marked as p corresponds to A2pk object. Similarly, each q block reflects the A2qk object. The blocks marked as ρ indicate that the respective column is fully transformed

for the teaching process is used, all neurons are able to update their weights at the same time. Due to that, multiplication of complexity by N_l is omitted. During the simulation the parameters n (layer input size) and N (layer's neuron count) have been adjusted to obtain the results presented in Figs. 8 and 9.

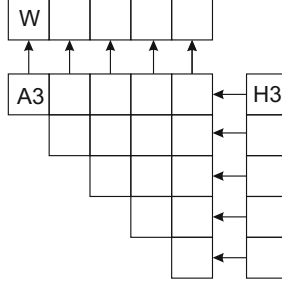


Fig. 6. The structure of the weight update system.

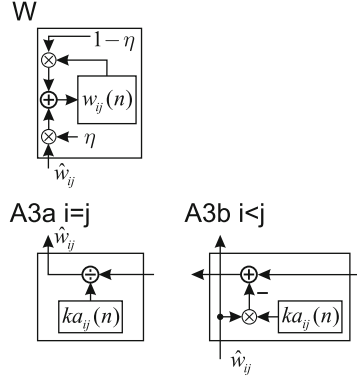


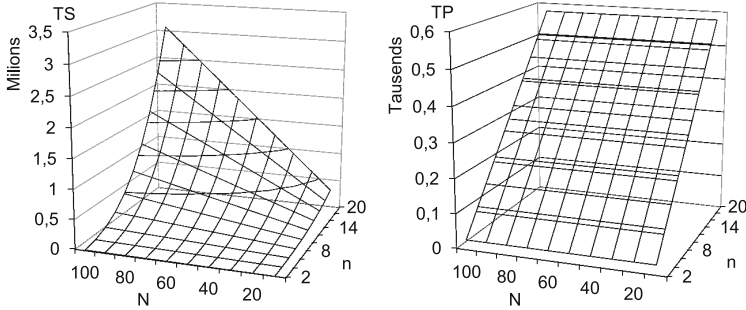
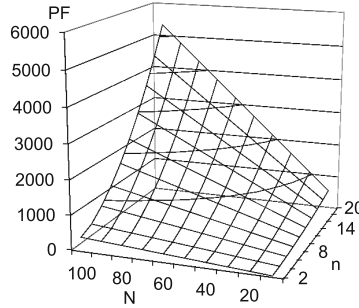
Fig. 7. Processing elements A3 and W structures. The only purpose of H3 objects is to provide values for further processing.

Table 1. Values of b depending on $x \in [x_{r_0}, x_{r_{end}}]$ for $y = 2x - b$.

x_{r_0}	$x_{r_{end}}$	b	x_{r_0}	$x_{r_{end}}$	b	x_{r_0}	$x_{r_{end}}$	b	x_{r_0}	$x_{r_{end}}$	b	x_{r_0}	$x_{r_{end}}$	b
2	3	3	202	229	16	776	835	29	1724	1815	42	3048	3165	55
4	7	4	230	266	17	836	895	30	1816	1905	43	3166	3287	56
8	14	5	267	298	18	896	959	31	1906	1998	44	3288	3405	57
15	23	6	299	338	19	960	1023	32	1999	2093	45	3406	3537	58
24	31	7	339	376	20	1024	1097	33	2094	2189	46	3538	3657	59
32	47	8	377	417	21	1098	1166	34	2190	2289	47	3658	3790	60
48	61	9	418	463	22	1167	1239	35	2290	2391	48	3791	3918	61
62	79	10	464	508	23	1240	1314	36	2392	2498	49	3919	4051	62
80	100	11	509	558	24	1315	1391	37	2499	2598	50	4052	4187	63
101	121	12	559	610	25	1392	1470	38	2599	2712	51	4188	4322	64
122	146	13	611	663	26	1471	1552	39	2713	2817	52	4323	4465	65
147	170	14	664	718	27	1553	1641	40	2818	2936	53	4466	4603	66
171	201	15	719	775	28	1642	1723	41	2937	3047	54	4604	4752	67

Table 2. Operation complexity of serial and parallel complete teaching processes for single layer l .

Operation	Serial	Parallel
$+/-$	$N_l \left(\frac{4n^3 + 15n^2 + 11n}{6} + 3 \right)$	$5n - 5$
$*/\div$	$N_l \left(\frac{8n^3 + 39n^2 + 25n}{6} + 5 \right)$	$12n - 13$
f/f'	$N_l (2n^2 - 2n + 1)$	$2n - 2$

**Fig. 8.** Computation complexity for serial (left) and parallel (right) implementations.**Fig. 9.** Performance factor achieved by parallel implementation.

7 Conclusion

The paper presents a full parallel QR decomposition process. The use of described structures can bring satisfying acceleration even for small networks (up to 10 neurons) and very significant acceleration for big networks (over 100 neurons per layer). The serial complexity is of order $\mathcal{O}(n^3 N_l)$ while the parallel one is of $\mathcal{O}(n)$. During the research the performance factor established the highest value of 5335 for $n = 20$ and $N_l = 100$. The trend of growth can be seen in Fig. 9. Parallelization of teaching algorithms seems to be a good direction for further optimization research. A similar parallel approach might be applied for other algorithms [1, 13, 18, 20]. The next step of the parallel Givens research is to

acquire an equation to get the value of b . In the near future also a momentum variant of Givens QR decomposition will be attempted as proposed in [5].

References

1. Starczewski, A.: A clustering method based on the modified RS validity index. In: Rutkowski, L., Korytkowski, M., Scherer, R., Tadeusiewicz, R., Zadeh, L.A., Zurada, J.M. (eds.) ICAISC 2013. LNCS, vol. 7895, pp. 242–250. Springer, Heidelberg (2013). doi:[10.1007/978-3-642-38610-7_23](https://doi.org/10.1007/978-3-642-38610-7_23)
2. Starczewski, A.: A new validity index for crisp clusters. *Pattern Anal. Appl.* (2015)
3. Starczewski, A., Krzyżak, A.: A modification of the silhouette index for the improvement of cluster validity assessment. In: Rutkowski, L., Korytkowski, M., Scherer, R., Tadeusiewicz, R., Zadeh, L.A., Zurada, J.M. (eds.) ICAISC 2016. LNCS, vol. 9693, pp. 114–124. Springer, Cham (2016). doi:[10.1007/978-3-319-39384-1_10](https://doi.org/10.1007/978-3-319-39384-1_10)
4. Bilski, J., Smoląg, J., Galushkin, A.I.: The parallel approach to the conjugate gradient learning algorithm for the feedforward neural networks. In: Rutkowski, L., Korytkowski, M., Scherer, R., Tadeusiewicz, R., Zadeh, L.A., Zurada, J.M. (eds.) ICAISC 2014. LNCS(LNAI), vol. 8467, pp. 12–21. Springer, Cham (2014). doi:[10.1007/978-3-319-07173-2_2](https://doi.org/10.1007/978-3-319-07173-2_2)
5. Bilski, J.: Momentum modification of the RLS algorithms. In: Rutkowski, L., Siekmann, J.H., Tadeusiewicz, R., Zadeh, L.A. (eds.) ICAISC 2004. LNCS, vol. 3070, pp. 151–157. Springer, Heidelberg (2004). doi:[10.1007/978-3-540-24844-6_18](https://doi.org/10.1007/978-3-540-24844-6_18)
6. Bilski, J.: *Struktury równoległe dla jednokierunkowych i dynamicznych sieci neuronowych*. Akademicka Oficyna Wydawnicza EXIT, Warszawa (2013)
7. Bilski, J., Kowalczyk, B., Żurada, J.M.: Application of the givens rotations in the neural network learning algorithm. In: Rutkowski, L., Korytkowski, M., Scherer, R., Tadeusiewicz, R., Zadeh, L.A., Zurada, J.M. (eds.) ICAISC 2016. LNCS(LNAI), vol. 9692, pp. 46–56. Springer, Cham (2016). doi:[10.1007/978-3-319-39378-0_5](https://doi.org/10.1007/978-3-319-39378-0_5)
8. Bilski, J., Wilamowski, B.M.: Parallel learning of feedforward neural networks without error backpropagation. In: Rutkowski, L., Korytkowski, M., Scherer, R., Tadeusiewicz, R., Zadeh, L.A., Zurada, J.M. (eds.) ICAISC 2016. LNCS(LNAI), vol. 9692, pp. 57–69. Springer, Cham (2016). doi:[10.1007/978-3-319-39378-0_6](https://doi.org/10.1007/978-3-319-39378-0_6)
9. Bilski, J., Smoląg, J., Żurada, J.M.: Parallel approach to the levenberg-marquardt learning algorithm for feedforward neural networks. In: Rutkowski, L., Korytkowski, M., Scherer, R., Tadeusiewicz, R., Zadeh, L.A., Zurada, J.M. (eds.) ICAISC 2015. LNCS(LNAI), vol. 9119, pp. 3–14. Springer, Cham (2015). doi:[10.1007/978-3-319-19324-3_1](https://doi.org/10.1007/978-3-319-19324-3_1)
10. Bilski, J., Rutkowski, L.: Numerically robust learning algorithms for feed forward neural networks. In: Rutkowski, L., Kacprzyk, J. (eds.) *Neural Networks and Soft Computing*. Advances in Soft Computing, pp. 149–154. Physica-Verlag, Heidelberg (2003)
11. Bilski, J., Smoląg, J.: Parallel approach to learning of the recurrent jordan neural network. In: Rutkowski, L., Korytkowski, M., Scherer, R., Tadeusiewicz, R., Zadeh, L.A., Zurada, J.M. (eds.) ICAISC 2013. LNCS(LNAI), vol. 7894, pp. 32–40. Springer, Heidelberg (2013). doi:[10.1007/978-3-642-38658-9_3](https://doi.org/10.1007/978-3-642-38658-9_3)
12. Werbos, J.: *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*. Harvard University (1974)

13. Nowicki, R.K., Nowak, B.A., Starczewski, J.T., Cpałka, K.: The learning of neuro-fuzzy approximator with fuzzy rough sets in case of missing features. In: 2014 International Joint Conference on Neural Networks (IJCNN), pp. 3759–3766, July 2014
14. Bilski, J., Rutkowski, L.: A fast training algorithm for neural networks. *IEEE Trans. Circ. Syst. Part II* **45**, 749–753 (1998)
15. Zalaśiński, M., Lapa, K., Cpałka, K.: New algorithm for evolutionary selection of the dynamic signature global features. In: Rutkowski, L., Korytkowski, M., Scherer, R., Tadeusiewicz, R., Zadeh, L.A., Zurada, J.M. (eds.) *ICAISC 2013. LNCS (LNAI)*, vol. 7895, pp. 113–121. Springer, Heidelberg (2013). doi:[10.1007/978-3-642-38610-7_11](https://doi.org/10.1007/978-3-642-38610-7_11)
16. Zalaśiński, M., Cpałka, K., Rakus-Andersson, E.: An idea of the dynamic signature verification based on a hybrid approach. In: Rutkowski, L., Korytkowski, M., Scherer, R., Tadeusiewicz, R., Zadeh, L.A., Zurada, J.M. (eds.) *ICAISC 2016. LNCS (LNAI)*, vol. 9693, pp. 232–246. Springer, Cham (2016). doi:[10.1007/978-3-319-39384-1_21](https://doi.org/10.1007/978-3-319-39384-1_21)
17. Zalaśiński, M., Cpałka, K., Hayashi, Y.: New method for dynamic signature verification based on global features. In: Rutkowski, L., Korytkowski, M., Scherer, R., Tadeusiewicz, R., Zadeh, L.A., Zurada, J.M. (eds.) *ICAISC 2014. LNCS*, vol. 8468, pp. 231–245. Springer, Cham (2014). doi:[10.1007/978-3-319-07176-3_21](https://doi.org/10.1007/978-3-319-07176-3_21)
18. Zalaśiński, M., Cpałka, K., Hayashi, Y.: A new approach to the dynamic signature verification aimed at minimizing the number of global features. In: Rutkowski, L., Korytkowski, M., Scherer, R., Tadeusiewicz, R., Zadeh, L.A., Zurada, J.M. (eds.) *ICAISC 2016. LNCS (LNAI)*, vol. 9693, pp. 218–231. Springer, Cham (2016). doi:[10.1007/978-3-319-39384-1_20](https://doi.org/10.1007/978-3-319-39384-1_20)
19. Mleczo, W.K., Kapuściński, T., Nowicki, R.K.: Rough deep belief network - application to incomplete handwritten digits pattern classification. In: Dregvaite, G., Damasevicius, R. (eds.) *ICIST 2015. CCIS*, vol. 538, pp. 400–411. Springer, Cham (2015). doi:[10.1007/978-3-319-24770-0_35](https://doi.org/10.1007/978-3-319-24770-0_35)
20. Nowak, B.A., Nowicki, R.K., Starczewski, J.T., Marvuglia, A.: The learning of neuro-fuzzy classifier with fuzzy rough sets for imprecise datasets. In: Rutkowski, L., Korytkowski, M., Scherer, R., Tadeusiewicz, R., Zadeh, L.A., Zurada, J.M. (eds.) *ICAISC 2014. LNCS (LNAI)*, vol. 8467, pp. 256–266. Springer, Cham (2014). doi:[10.1007/978-3-319-07173-2_23](https://doi.org/10.1007/978-3-319-07173-2_23)
21. Tadeusiewicz, R.: *Sieci Neuronowe*. Akademicka Oficyna Wydawnicza, Warszawa (1993)
22. Nowak, B.A., Nowicki, R.K.: Learning in rough-neuro-fuzzy system for data with missing values. In: Wyrzykowski, R., Dongarra, J., Karczewski, K., Waśniewski, J. (eds.) *PPAM 2011. LNCS*, vol. 7203, pp. 501–510. Springer, Heidelberg (2012). doi:[10.1007/978-3-642-31464-3_51](https://doi.org/10.1007/978-3-642-31464-3_51)
23. Knop, M., Kapuscinski, T., Mleczo, W.K.: Video key frame detection based on the restricted boltzmann machine. *J. Appl. Math. Comput. Mech.* **14**(3), 49–58 (2015)
24. Wozniak, M., Polap, D., Nowicki, R.K., Napoli, C., Pappalardo, G., Tramontana, E.: Novel approach toward medical signals classifier. In: 2015 International Joint Conference on Neural Networks (IJCNN), pp. 1–7, July 2015

Artificial Intelligence and Soft Computing
16th International Conference, ICAISC 2017, Zakopane,
Poland, June 11-15, 2017, Proceedings, Part I
Rutkowski, L.; Korytkowski, M.; Scherer, R.;
Tadeusiewicz, R.; Zadeh, L.A.; Zurada, J.M. (Eds.)
2017, XXIV, 776 p. 293 illus., Softcover
ISBN: 978-3-319-59062-2