Multi-phase Adaptive Competitive Learning Neural Network for Clustering Big Datasets

Mohamed G. Mahdy[1](✉), Ahmed R. Abas[1], and Tarek M. Mahmoud[2]

[1] Department of Computer Science, Faculty of Computers and Informatics, Zagazig University, Zagazig, Egypt
arabas@zu.edu.eg

[2] Department of Computer Science, Faculty of Science, Minia University, Minia, Egypt
d.tarek@mu.edu.eg

Abstract. The Competitive Learning Neural Network (CLNN) algorithm is used for the classification of numerical datasets. The Adaptive Competitive Learning Neural Network (ACLNN) algorithm is a modification of the CLNN algorithm and produces high accuracy results with small and moderate sizes of datasets. However, it has high time complexity with big datasets, and the accuracy of the data clustering is low. To overcome these drawbacks, a Multi-phase Adaptive Competitive Learning Neural Network (MACLNN) is proposed. The proposed algorithm consists of three phases. The first phase in the proposed algorithm splits big datasets into equal partitions called sub-datasets. This phase aims to keep the dataset's characteristics and speeding up the clustering process. The second phase in the proposed algorithm uses the sub-datasets as input data to the ACLNN algorithm. This phase aims to determine the optimal number of clusters. To speed up this phase, a parallel processing technique is used. The last phase in the proposed algorithm uses the extensive dataset and the optimal number of clusters determined from the second phase as input to the ACLNN algorithm. This phase aims to determine the clustering id for every data object in the input dataset. To assess the effectiveness of the proposed algorithm, twelve experimental datasets are used. The results obtained are compared to those obtained by the ACLNN algorithm. Evaluation of the proposed algorithm on big datasets shows that it outperforms the ACLNN algorithm on both the clustering accuracy and the running time.

Keywords: Adaptive Competitive Learning · Neural network · Clustering · Big datasets · Multi-phases adaptive competitive learning · Normalized Mutual Information · Parallel processing

1 Introduction

The Competitive Learning Neural Network (CLNN) algorithm is used for clustering and classification analysis [1–5]. It consists of an input layer and a single competitive learning output layer. Examples of the CLNN algorithms are Self-Organizing Map (SOM) [6–8] and Learning Vector Quantization (LVQ) [6, 7]. In the CLNN algorithm, each input node is fully connected to the output neurons [9–13]. Several algorithms are proposed to determine the number of neurons in the output layer of the CLNN algorithm that are considered the number of clusters in the input dataset. One algorithm compares the differences among the synaptic weight vectors of the output neurons to a specified value, and if the difference between the weight vectors of two neurons is less than or equal to the specified value, then one of these neurons is removed. Otherwise, the number of the output neurons is increased by one [14]. Another algorithm removes the dead neurons that have lost the competition in representing the input feature vectors after convergence. However, this algorithm is sensitive to the selection of the learning rate.

The Adaptive Competitive Learning Neural Network (ACLNN) algorithm is a modification of the CLNN algorithm that generates high accuracy results for small and moderate-sized datasets.

Parallel processing usually uses GPU cores instead of CPU cores because they are fast and large [15, 16]. The feedforward neural network is the only neural network whose training function is parallelized to GPU cores [17]. On the other hand, the Competitive Learning Neural Network (CLNN) training function is sequential and cannot be applied to GPU cores [17].

In this paper, a Multi-Phase Adaptive Competitive Learning Neural Network has been proposed to cluster big input datasets. The first phase of the proposed algorithm divides big datasets into equal partitions (sub-datasets). In the second phase, the ACLNN algorithm is implemented using a parallel processing technique. As a result of this phase, the number of clusters is determined. The last phase uses this number of clusters for clustering the original big dataset.

The main contributions of this paper are epitomized as follows:

- Creating the MACLNN for determining the number of clusters and clustering of input big datasets.
- Using multiple samples of the input dataset with parallel processing to reduce the running time of the MACLNN.

This paper is structured as follows; related work is presented in Sect. 2. The proposed MACLNN is presented in Sect. 3. Performance evaluation of the MACLNN is presented in Sect. 4. The results are discussed in Sect. 5. Finally, the conclusion is presented in Sect. 6.

2 Related Work

The Adaptive Competitive Learning Neural Network (ACLNN) algorithm was proposed to determine the number of clusters and cluster an input small dataset. The ACLNN algorithm uses the Adaptive Competitive Learning (ACL) criterion for determining the optimum number of output neurons [18, 19]. Figure 1 shows the steps of the ACLNN algorithm.

The ACL criterion is based on the theory that the best cluster structure is composed of balanced, dense, and well-separated clusters with the least number of parameters to be calculated. The ACLNN algorithm proved efficient in identifying the number of clusters and clustering small datasets [18, 19]. Due to its sequential nature of running, the

```
Program CLNN = ACLNN (data)
Step 1.   Normalize the values of each input data feature to range from 0 to 1.
Step 2.   Within-cluster variations, product of the relative weights of clusters
          and the number of neurons should be minimized during optimization.
Step 3.   Start with a CLNN that has a large number of neurons k = kmax and
          empty Best CLNN.
Step 4.   Do {
          Step 5.   Use learning with conscience to learn weight vectors of the
                    neurons in the current CLNN.
          Step 6.   Present the input feature vectors to the trained CLNN and
                    obtain the corresponding output vectors and cluster indexes.
          Step 7.   Compute the average within-cluster variations E(k) and the
                    relative weights of clusters πj, j = 1: k.
          Step 8.   Compute the ACL(k) criterion value for the current CLNN.
          Step 9.   If (Best CLNN is Empty or ACL(k) < Best
                    CLNN.ACLNN) then save the current weight vectors and
                    ACL(k) as Best CLNN.
          Step 10.  Else create a new CLNN with k=k–1
          Step 11.  end If
Step 12.  } while (k ≥ 1)
Step 13.  Use the Best CLNN as the optimal CLNN for clustering the input da-
          taset.
Step 14.  Stop.
```

Fig. 1. The steps of the ACLNN

ACLNN algorithm has a high time complexity. Therefore it takes a long-running time, especially when used with big datasets. Also, the ACLNN algorithm doesn't properly use hardware recourses provided by modern multi-core processors. Parallel processing allows more hardware resources to be used and therefore decreases the running time [15, 20]. Parallel processing requires the algorithm to be parallelized to be divided into many independent tasks to avoid problems such as deadlock, starvation, and race condition [20].

Several research works in the literature implement parallel computationally expensive Artificial Neural Networks (ANNs) to be run on parallel or distributed computing systems [21]. Special-purpose hardware, which is referred to as neurohardware or neurocomputer, is adopted to improve the training speed of the ANN [22]. This hardware is fast and efficient. However, it offers little flexibility and scalability [23]. After the 1990s, designing parallel neural networks over general-purpose architectures, such as parallel computing models or grid computing models, became the mainstream [24]. These systems are mostly implemented on clusters of multi-processor computers. However, this paper makes few efforts to manage large datasets. It usually focuses on studying how to parallelize neural network training and only performs experiments with several thousands of training samples and Mega-byte-class of data size [25].

3 The Proposed MACLNN

The proposed MACLNN consists of three phases. Figure 2 shows the steps of this algorithm.

```
($K_{max}$, Clustering Results, NMI) = Program MACLNN (data)
Step 0.  Normalize the values of each input data feature to range from 0 to 1
Step 1.  Divide the input big dataset into N equal sub-datasets. (phase 1)
Step 2.  Open parallel pool (phase 2)
Step 3.  For loop I = 1 to N in parallel
            Step 4. Apply the ACLNN algorithm on each sub-dataset and determine the number of clusters (K).
            Step 5. Save the result.
Step 6.  End of for Loop.
Step 7.  Close parallel pool
Step 8.  Select Maximum K ($k_{max}$) produced from steps 4 & 5.
Step 9.  Use ACLNN algorithm ($K_{max}$ =maximum k, original big dataset). (Phase 3)
Step 10. Save the clustering id for every data object in the input big dataset and calculate the NMI value of it.
Step 11. Return the $K_{max}$, NMI value, and its clustering results.
```

Fig. 2. The steps of the MACLNN

3.1 The Pre-processing Phase (Phase 1)

All features of the input dataset are normalized to be from 0 to 1. Then, the dataset is divided into N sub-datasets of equal sizes while preserving the dataset's characteristics.

3.2 Phase 2

The created N sub-datasets are used as input to N separate copies of the Adaptive Competitive Learning Neural Network (ACLNN) algorithm, working in parallel (multiple CPU cores). During this step, the number of clusters is determined. In our implementation, we choose the maximum number of clusters (K_{max}) as the optimal value of clusters and denote it by K_{max}.

3.3 Phase 3

This phase's input is the original big dataset and the maximum number of clusters (K_{max}) produced by phase 2. The ACLNN algorithm is running N times, and the output is stored in a structure. The clustering id for every data object in the big input dataset and the maximum Normalized Mutual Information (NMI) is obtained.

4 Experiments and Results

The performance of the MACLNN is compared to the performance of the ACLNN in identifying the number of clusters and clustering input datasets of different sizes. The Neural Network Toolbox and deep Learning Toolbox [6, 7] in MATLAB R2018a platform are used to implement and compare both algorithms. In this evaluation study, datasets have various properties of clustering structures such as data sparsity, cluster sizes, and cluster separation. Both algorithms are run on the DELL G5 15 Lap-top that has an i7-intel processor, a 2.20 GHz CPU, and 16 GB RAM. Windows 10 is the operating system used on the laptop.

Section 4.1 describes the datasets used in experiments. The measures used to evaluate the clustering performance of both algorithms are provided in Sect. 4.2.

4.1 Datasets

Tables 1 and 2 show a description of the big datasets of different cluster structure properties used in the experiments to test the performance of the MACLNN. Standard real datasets are chosen from the UCI repository [26]. Their sizes are extended by replication [27] of data 100 times for the Iris, the Seed, and the Breast Cancer dataset, as shown in Table 1. Generated artificial datasets composed of Gaussian-shaped clusters with different properties of clustering structures regarding data sparsity, size of clusters, and cluster separation are shown in Table 2.

Table 1. The description of the real datasets

Date Set	Size	No. of Features	Sizes of Classes	Clustering Characteristics

Iris_100	15000	4	1(5000)
			2(5000)
			3(5000)
				- Poorly separated
				- Balanced data

Seed_100	21000	7	1(7000)
			2(7000)
				- Poorly separated
				- Balanced data

Breast cancer_100	47800	9	1(23900)
			2(23900)
				- Sparse
				- Poorly separated
				- Balanced data

Table 2. The description of the generated datasets

DataSet	Size	No. of Features	Sizes of Classes	Centers of Gaussian-shaped Clusters	Covariance-Matrices	Clustering Characteristics
			1(3000)	μ1= [2, 2, 2, 2, 2, 2, 2, 2, 2, 2] T		
			2(3000)	μ2= [6, 2, 2, 2, 6, 6, 2, 2, 6, 6] T		- Sparse
Frist DataSet	15000	10	3(3000)	μ3= [2, 6, 6, 6, 2, 2, 6, 6, 2, 2] T	$\sum = 0.5*I_{10}$	- Well separated
			4(3000)	μ2= [2, 2, 2, 2, 2, 2, 2, 2, 2, 2] T		- Balanced data
			5(3000)	μ5= [6, 6, 6, 6, 6, 6, 6, 6, 6, 6] T		
			1(5000)	μ_1= [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]^T		- Sparse
Second DataSet	15000	10	2(5000)	μ2= [-2, -2, -2, -2, -2, -2, -2, -2, -2, -2] T	$\sum = I_{10}$	- Poorly separated
			3(5000)	μ3= [2, 2, 2, 2, 2, 2, 2, 2, 2, 2] T		- Balanced data
			1(3000)	μ1= [2, 2, 2, 2, 2, 2, 2, 2, 2, 2] T		
			2(4500)	μ2= [6 2, 2, 2, 6, 6, 2, 2, 6, 6] T		- Sparse
Third DataSet	15000	10	3(1500)	μ3= [2, 6, 6, 6, 2, 2, 6, 6, 2, 2] T	$\sum = 0.5*I_{10}$	- Well separated
			4(3000)	μ2= [2, 2, 2, 2, 2, 2, 2, 2, 2, 2] T		- Imbalanced data
			5(3000)	μ5= [6, 6, 6, 6, 6, 6, 6, 6, 6, 6] T		
			1(4500)	μ_1= [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]^T		- Sparse
Fourth DataSet	15000	10	2(6000)	μ2= [-2, -2, -2, -2, -2, -2, -2, -2, -2, -2] T	$\sum = I_{10}$	- Poorly separated
			3(4500)	μ3= [2, 2, 2, 2, 2, 2, 2, 2, 2, 2] T		- Imbalanced data
			1(80000)	μ1= [2, 2, 2, 2, 2, 2, 2, 2, 2, 2] T		
			2(80000)	μ2= [6, 2, 2, 2, 6, 6, 2, 2, 6, 6] T		- Condensed
Fifth DataSet	60000	10	3(80000)	μ3= [2, 6, 6, 6, 2, 2, 6, 6, 2, 2] T	$\sum = 0.5*I_{10}$	- Well separated
			4(80000)	μ2= [2, 2, 2, 2, 2, 2, 2, 2, 2, 2] T		- Balanced data
			5(80000)	μ5= [6, 6, 6, 6, 6, 6, 6, 6, 6, 6] T		
			1(150000)	μ_1= [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]^T		- Condensed
Sixth DataSet	60000	10	2(150000)	μ2= [-2, -2, -2, -2, -2, -2, -2, -2, -2, -2] T	$\sum = I_{10}$	- Poorly separated
			3(150000)	μ3= [2, 2, 2, 2, 2, 2, 2, 2, 2, 2] T		- Balanced data
			1(80000)	μ1= [2, 2, 2, 2, 2, 2, 2, 2, 2, 2] T		
			2(40000)	μ2= [6, 2, 2, 2, 6, 6, 2, 2, 6, 6] T		- Condensed
Seventh DataSet	60000	10	3(120000)	μ3= [2, 6, 6, 6, 2, 2, 6, 6, 2, 2] T	$\sum = 0.5*I_{10}$	- Well separated
			4(100000)	μ2= [2, 2, 2, 2, 2, 2, 2, 2, 2, 2] T		- Imbalanced data
			5(60000)	μ5= [6, 6, 6, 6, 6, 6, 6, 6, 6, 6] T		
			1(180000)	μ_1= [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]^T		- Condensed
Eighth DataSet	60000	10	2(157500)	μ2= [-2, -2, -2, -2, -2, -2, -2, -2, -2, -2] T	$\sum = I_{10}$	- Poorly separated
			3(112500)	μ3= [2, 2, 2, 2, 2, 2, 2, 2, 2, 2] T		- Imbalanced data

4.2 Measure of Performance

The Normalized Mutual Information (NMI) is a measure that is used for evaluating the performance of clustering [11, 28]. The NMI value ranges from 0 (perfect mismatch) to 1 (perfect match). It reflects the amount of information generated by the clusters.

The mutual information between X and Y is defined as

$$I(X; Y) = \sum\nolimits_{i=1}^{m} \sum\nolimits_{j=1}^{k} P_{ij} \log_2 \left(P_{ij}/P_i P_j \right) \tag{1}$$

P_{ij} is the probability that a member of cluster j belongs to class i, P_i is the probability of class i and P_j is the probability of cluster j.

$$NMI(X; Y) = \frac{I(X; Y)}{\sqrt{H(X)H(Y)}} \tag{2}$$

Where H(X) and H(Y) denote the entropy of X and Y.

Table 3, Figs. 3 and 4 show a performance comparison of the MACLNN and the ACLNN using the datasets. The performances of both algorithms are assessed using the maximum and standard deviation of the NMI values, and the maximum and standard deviation of the number of clusters (K_{MAX}) corresponding to the optimal CLNN algorithm selected by both algorithms according to the ACL criterion value provided by ten different experiments. Every experiment utilizes various random values for the initialization of the neuron weight vectors of the CLNN algorithm. This replication of the experiments reduces the influence of initialization values on output results.

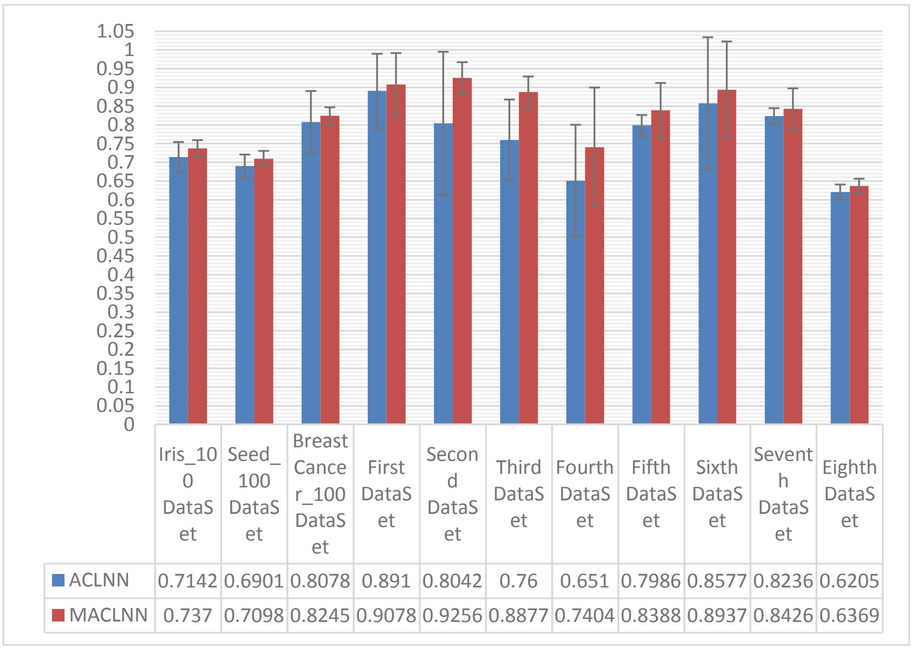
5 Discussion of Results

Table 3, Figs. 3 and 4 show that the performance of the MACLNN is better than the ACLNN algorithm in the identification of the number of clusters (approximately exact real k) with all datasets used. Also, they show that the NMI obtained by the MACLNN increases by 1.5% to 2% more than the ACLNN. Finally, they show that the MACLNN is from 1.5 to 5.95 times faster than the ACLNN algorithm.

There are two reasons for the fast performance of the MACLNN compared to the ACLNN. First, working in parallel in phase 2 of the MACLNN and using small sub-datasets to determine the maximum number of clusters in the input big dataset rather than working sequentially on the whole dataset ACLNN. Second, using the maximum number of clusters obtained from phase 2 of the MACLNN, which is a small number, in phase 3 to determine the optimal number of clusters and clustering the input dataset instead of the large number used in the ACLNN algorithm, which is 16.

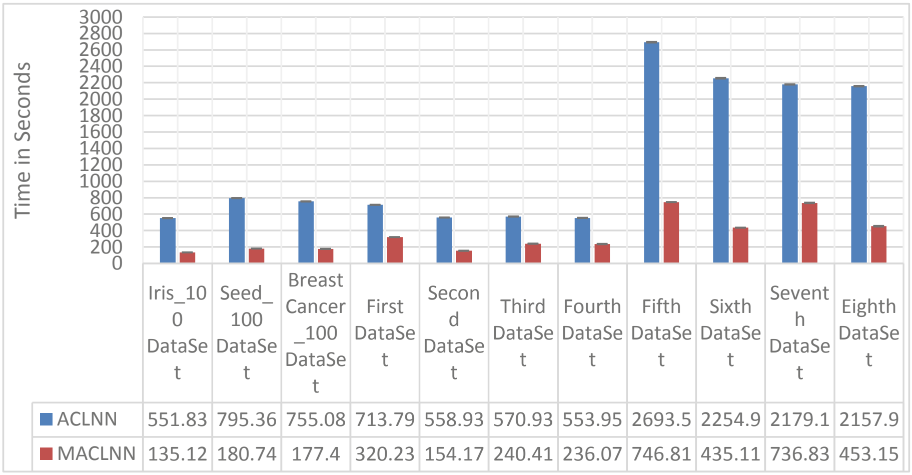
Table 3. Performance comparison of the MACLNN and the ACLNN

Data Sets	K_opt				NMI				Time in Seconds				Speed UP %
	Maximum		STD		Maximum		STD		AVG		STD		
	ACLNN	MACLNN	ACLNN	MACLNN	ACLNN	MACLNN	ACLNN	MACLNN	ACLNN	MACLNN	ACLNN	MACLNN	
Iris_100 DataSet	3	3	0	0	0.7142	**0.737**	0.0402	0.0226	551.8349	135.1215	2.4415	0.5478	4.084
Seed_100 DataSet	3	3	0	0	0.6901	**0.7098**	0.031	0.021	795.3635	180.7427	1.8715	0.7539	4.4005
Breast Cancer_100 DataSet	2	2	0.6325	0	0.8078	**0.8245**	0.083	0.0228	755.0823	177.4038	2.4654	2.2374	4.7015
First DataSet	4	5	0.5164	0.483	0.891	**0.9078**	0.0992	0.0843	713.7943	320.2312	1.5715	1.2418	2.229
Second DataSet	3	3	0.4216	0.483	0.8042	**0.9256**	0.1913	0.042	558.9301	154.1703	2.0477	2.2821	3.6254
Third DataSet	4	5	0.527	0	0.76	**0.8877**	0.108	0.0418	570.9275	240.4092	2.5925	2.3281	2.3748
Fourth DataSet	2	3	0.4216	0.527	0.651	**0.7404**	0.1495	0.1594	553.9457	236.0731	2.5084	3.2538	2.3465
Fifth DataSet	4	5	0	0.4216	0.7986	**0.8388**	0.028	0.0735	2693.5	746.8128	4.5527	1.1744	3.6067
Sixth DataSet	3	3	0.4216	0.3162	0.8577	**0.8937**	0.1766	0.1293	2254.9	435.1103	4.82235	0.625	5.1824
Seventh DataSet	4	5	0	0.3162	0.8236	**0.8426**	0.0213	0.0552	2179.1	736.8348	2.67201	3.73052	2.9574
Eighth DataSet	2	2	0	0	0.6205	**0.6369**	0.0206	0.0196	2157.9	453.1465	3.69621	4.78915	4.7622



	Iris_100 DataSet	Seed_100 DataSet	Breast Cancer_100 DataSet	First DataSet	Second DataSet	Third DataSet	Fourth DataSet	Fifth DataSet	Sixth DataSet	Seventh DataSet	Eighth DataSet
🟦 ACLNN	0.7142	0.6901	0.8078	0.891	0.8042	0.76	0.651	0.7986	0.8577	0.8236	0.6205
🟥 MACLNN	0.737	0.7098	0.8245	0.9078	0.9256	0.8877	0.7404	0.8388	0.8937	0.8426	0.6369

Fig. 3. The average and the standard deviation of the NMI



	Iris_100 DataSet	Seed_100 DataSet	Breast Cancer_100 DataSet	First DataSet	Second DataSet	Third DataSet	Fourth DataSet	Fifth DataSet	Sixth DataSet	Seventh DataSet	Eighth DataSet
🟦 ACLNN	551.83	795.36	755.08	713.79	558.93	570.93	553.95	2693.5	2254.9	2179.1	2157.9
🟥 MACLNN	135.12	180.74	177.4	320.23	154.17	240.41	236.07	746.81	435.11	736.83	453.15

Fig. 4. The average and the standard deviation of running time

6 Conclusion

In this paper, the Multi-phase Adaptive Competitive Learning Neural Network (MACLNN) is proposed for clustering big datasets. The proposed algorithm uses multiple equal sub-datasets of the input big dataset, and parallel processing through using multiple CPU-cores processors. During the phases of this algorithm, the maximum number of clusters is determined. This number is used to determine the clustering id of each object of the original big dataset. A performance evaluation study was carried out and shows that the proposed algorithm outperforms the performance of the ACLNN algorithm, which was previously proposed in the literature, in terms of running time and clustering accuracy. The NMI values of clustering structures obtained from the MACLNN are from 1.5% to 2.5% higher than the NMI values of clustering structures obtained from the ACLNN algorithm. In addition, the speedup of the performance obtained by the MACLNN increases with the increase in the size of the input dataset.

In the future, we plan to combine more clustering algorithms and compare the cluster accuracy with the MACLNN. Also, we can try different datasets with more dimensions that are collected from different resources like images, and posts from social media.

References

1. Oyedotun, O.K., Khashman, A.: Banknote recognition: investigating processing and cognition framework using competitive neural network. Cogn. Neurodyn. **11**, 67–79 (2017). https://doi.org/10.1007/s11571-016-9404-2
2. Li, W., Gu, Y., Yin, D., Xia, T., Wang, J.: Research on the community number evolution model of public opinion based on stochastic competitive learning. IEEE Access **8**, 46267–46277 (2020). https://doi.org/10.1109/ACCESS.2020.2978522
3. Zidan, M., Abdel-Aty, A.-H., El-shafei, M., Feraig, M., Al-Sbou, Y., Eleuch, H., Abdel-Aty, M.: Quantum classification algorithm based on competitive learning neural network and entanglement measure. Appl. Sci. **9**, 1277 (2019). https://doi.org/10.3390/app9071277
4. Qu, L., Zhao, Z., Wang, L., Wang, Y.: Efficient and hardware-friendly methods to implement competitive learning for spiking neural networks. Neural Comput. Appl. **32**, 13479–13490 (2020). https://doi.org/10.1007/s00521-020-04755-4
5. Li, T., Kou, G., Peng, Y., Shi, Y.: Classifying with adaptive hyper-spheres: an incremental classifier based on competitive learning. IEEE Trans. Syst. Man Cybern. Syst. **50**, 1218–1229 (2020). https://doi.org/10.1109/TSMC.2017.2761360
6. Beale, M.H., Hagan, M.T., Demuth, H.B.: Neural Network Toolbox TM User's Guide R2017b. Mathworks Inc. (2017)
7. Beale, M.H., Hagan, M.T., Demuth, H.B.: Deep Learning Toolbox User's Guide. Mathworks Inc., Herborn (2018)
8. Wickramasinghe, C.S., Amarasinghe, K., Manic, M.: Deep self-organizing maps for unsupervised image classification. IEEE Trans. Ind. Inform. **15**, 5837–5845 (2019). https://doi.org/10.1109/TII.2019.2906083
9. Kohonen, T.: Self-organization and Associative Memory. Springer, Heidelberg (2012)
10. Fukunaga, K.: Introduction to statistical pattern recognition. Elsevier (2013)
11. Du, K.L., Swamy, M.N.S.: Neural networks and statistical learning, Second edn. Springer, London (2019). https://doi.org/10.1007/978-1-4471-7452-3
12. Ripley, B.D.: Pattern Recognition and Neural Networks. Cambridge University Press, Cambridge (2007)

13. Bishop, C.M.: Pattern Recognition and Machine Learning. Springer, Heidelberg (2006)
14. Budura, G., Botoca, C., Miclău, N.: Competitive learning algorithms for data clustering. Facta Univ. Electron. Energ. **19**, 261–269 (2006)
15. Dinkelbach, H.Ü., Vitay, J., Beuth, F., Hamker, F.H.: Comparison of GPU-and CPU-implementations of mean-firing rate neural networks on parallel hardware. Netw. Comput. Neural Syst. **23**, 212–236 (2012). https://doi.org/10.3109/0954898X.2012.739292
16. Li, X., Zhang, G., Li, K., Zheng, W.: Deep learning and its parallelization. In: Big Data Princ. Paradig., pp. 95–118. Elsevier Inc. (2016). https://doi.org/10.1016/B978-0-12-805394-2.00004-0
17. Ploskas, N., Samaras, N.: GPU Programming in MATLAB. Morgan Kaufmann (2016)
18. Abas, A.R.: Adaptive competitive learning neural networks. Egypt. Informatics J. **14**, 183–194 (2013). https://doi.org/10.1016/j.eij.2013.08.001
19. Abas, A.R.: On determining efficient finite mixture models with compact and essential components for clustering data. Egypt. Informatics J. **14**, 79–88 (2013). https://doi.org/10.1016/j.eij.2013.02.002
20. C. Mathworks, Parallel Computing Toolbox TM User's Guide R 2018 a (2018)
21. Hidalgo Espinoza, S.H.: Intrusion Detection in Web Systems Using Deep Learning Techniques, Universidad de Investigación de Tecnología Experimental Yachay (2019)
22. Serbedzija, N.B.: Simulating artificial neural networks on parallel architectures. Comput. (Long. Beach. Calif) **29**, 56–63 (1996)
23. Heard, M., Ford, J., Yene, N., Straiton, B., Havanas, P., Guo, L.: Advancing the neurocomputer. Neurocomputing **284**, 36–51 (2018). https://doi.org/10.1016/j.neucom.2018.01.021
24. Chu, C.-T., Kim, S.K., Lin, Y.-A., Yu, Y., Bradski, G., Olukotun, K., Ng, Y.: Map-reduce for machine learning on multicore. In: Advances Neural Information Processing System, pp. 281–288 (2007)
25. Kharbanda, H., Campbell, R.H.: Fast neural network training on general purpose computers. in: Proceedings of International Conference High Performance Computing (2011)
26. Dua, D., Graff, C.: {UCI} Machine Learning Repository (2017). https://archive.ics.uci.edu/ml
27. Ilager, S., Prasad, P.S.V.S.S.: Scalable mapreduce-based fuzzy min-max neural network for pattern classification, In: ACM International Conference Proceeding Series. Association for Computing Machinery (2017). https://doi.org/10.1145/3007748.3007776
28. Amelio, A., Tagarelli, A.: Data mining: clustering, in: Encyclopedia Bioinformatics Computational Biology ABC Bioinformatics, pp. 437–448. Elsevier (2018). https://doi.org/10.1016/B978-0-12-809633-8.20489-5