A modified neuro-fuzzy counterpropagation network and its fast adaptive learning

Sergiy Popov[1], Iryna Pliss[1], Olha Chala[1] and Olexii Holovin[2]

[1] *Kharkiv National University of Radio Electronics, 14 Nauky av., Kharkiv, 61166, Ukraine*
[2] *Central Scientific Research Institute of Armament and Military Equipment of the Armed Forces of Ukraine, Kyiv, 03049, Ukraine*

Abstract
A neuro-fuzzy counterpropagation network is introduced that employs a modified fuzzy C-means clustering procedure in an online mode, enhancing both learning rate and accuracy while maintaining the same simple architecture as traditional CPN networks. This modification allows handling of overlapping classes, when an observation can belong to multiple classes simultaneously. Consequently, several output layer neurons can be activated at once. An optimized algorithm is introduced for the output layer tuning with a better control over its filtering and following characteristics through the use of a special adjustable parameter. Experiments demonstrate that this innovative approach outperforms traditional counterpropagation networks in various performance metrics.

Keywords
Counterpropagation network, neuro-fuzzy network, overlapping classes, increased learning rate, short training set

1. Introduction

In recent years, artificial neural networks (ANN) have become a popular solution to various information processing challenges. These include tasks such as pattern recognition (classification), clustering, and forecasting (extrapolation). The success of ANNs can be attributed to their ability to approximate complex functions (universal approximation properties) and learn by adjusting their parameters based on optimization procedures.

Deep neural networks (DNNs), a subset of ANNs, have demonstrated remarkable results in solving numerous data analysis problems. However, DNNs also have significant drawbacks. One major limitation is the requirement for large amounts of training data, which may not always be available. Additionally, DNNs can be slow during parameter adjustment in multi-epoch learning mode. DNNs also face challenges when tackling real-time data stream mining tasks under conditions of non-stationarity and limited input information. Similar limitations apply to classic multilayer perceptron (MLP) models trained using the error backpropagation procedure.

It is worth noting that classic radial basis function networks (RBFN) [1, 2] exhibit a higher learning rate but may encounter issues related to the “curse of dimensionality” as the number of input signals increases.

In today’s data-driven world, there is a growing need for neural networks that can efficiently handle data stream mining tasks in online mode with limited training data. Among various neural network models, the counterpropagation neural network (CPN), introduced by R. Hecht-Nielsen [3-5], stands out as a viable solution despite its architectural simplicity.

Advantages of CPNs:

- High learning rate: CPNs are known for their ability to learn quickly.
- Simple architecture: only two layers formed by simple nodes, CPNs offer computational efficiency.

[1]CMIS-2025: Eighth International Workshop on Computer Modeling and Intelligent Systems, May 5, 2025, Zaporizhzhia, Ukraine

serhii.popov@nure.ua (S. Popov); iryna.pliss@nure.ua (I. Pliss); olha.chala@nure.ua (O. Chala); a_a_golovin@ukr.net (O. Holovin)
0000-0002-1274-5830 (S. Popov); 0000-0001-7918-7362 (I. Pliss); 0000-0002-7603-1247 (O. Chala); 0000-0003-4662-4559 (O. Holovin)

However, there is an inherent trade-off. While CPNs excel in learning rate, their approximation properties – the ability to model complex functions – are inferior compared to traditional MLPs and RBFNs, and of course modern DNNs, which are generally more powerful in function approximation.

Despite these limitations, ongoing research focuses on enhancing the approximation capabilities of CPNs while maintaining their high learning and processing rates. These efforts aim to bridge the gap between performance and efficiency without compromising on speed. Recent applications of CPNs include but are not limited to classification [6-9], prediction [10], parameter identification [11], structural optimization [12], extreme learning machine optimization [13], digital image watermarking [14], navigation systems development [15] and others. One promising direction for improvement involves integrating hybrid systems of computational intelligence [16]. Specifically, neuro-fuzzy approaches, which combine neural networks with fuzzy logic, offer a potential solution. By leveraging these methods [2, 17, 18] it may be possible to enhance the characteristics of CPNs. In conclusion, while CPNs present unique challenges compared to more sophisticated neural architectures like DNNs and MLPs, ongoing research explores innovative solutions that could unlock their full potential.

2. Counterpropagation network basics

From a theoretical point of view, the counterpropagation network is intended for restoring the nonlinear mapping $y=F(x)$ (forward-only CPN architecture shown in Fig. 1 is sufficient), as well as the inverse mapping $x=F^{-1}(y)$ (full CPN architecture is required, see Fig. 2), i.e. identifying a nonlinear transform

$$F:X \to Y\left(R^n \to R^m\right)$$

from the training samples $x(1), y(1), \ldots, x(k), y(k), \ldots, x(N), y(N)$, where $x(k)=\left(x_1(k), \ldots, x_i(k), \ldots, x_n(k)\right)^T \in R^n$, $y(k)=\left(y_1(k), \ldots, y_i(k), \ldots, y_m(k)\right)^T \in R^m$, $k=1,2,\ldots,N$ is the observation index in the dataset, or the index of the current discrete time, if the data is being processed in online mode.

CPN contains two layers of neurons: the first hidden layer, called the T. Kohonen layer, and the output layer, called the S. Grossberg layer. In this paper, we will focus on the forward-only architecture, but the proposed methods are equally applicable to the full CPN architecture as well.

The input signals $x(k)$ arrive sequentially from the receptive layer to the first hidden layer, which is usually a Kohonen's self-organizing map (SOM) [19, 20] designed to solve the crisp clustering problem, i.e. dividing the data set into h non-overlapping classes/clusters in the self-learning mode. SOM implements the following "Winner Takes All" mapping

$$u_l(k)=\begin{cases} ¿\,1, if\ w_l^K(k)\, is\, a\, winner, i.e. \left\|x(k)-w_l^K(k)\right\| \le \left\|x(k)-w_i^K(k)\right\|\ \forall\, i=1,2,\ldots,h \\ ¿\,0\ otherwise \end{cases} \tag{1}$$

where $u(k)=\left(u_1(k), \ldots, u_l(k), \ldots, u_h(k)\right)^T$, $W^K=\left\{w_{li}^K\right\}-(h \times n)$ tuned matrix of synaptic weights that define centroids of the clusters.

The Kohonen layer learning is based on the same "Winner Takes All" (WTA) principle, when only one winning neuron is tuned at each iteration $k=1,2,\ldots$ When observation $x(k)$ is received, the closest to $x(k)$ neuron is determined (in the Euclidean metrics sense), which is called a "winner" (1). Then only this "winner" neuron's vector of weights $w_l^K(k)$ is being tuned according to the rule:

$$w_l^K(k+1)=\begin{cases} ¿\,w_l^K(k)+\eta_K(k)\left(x(k)-w_l^K(k)\right), if\ w_l^K(k)\, is\, a\, winner, \\ ¿\,w_l^K(k)\ otherwise \end{cases} \tag{2}$$

(here $0<\eta_K(k)<1$ is the Kohonen layer learning rate parameter, which is usually chosen empirically). Note also that when $\eta_K(k)=k^{-1}$ procedure (2) calculates the arithmetic mean (centroid) of the lth cluster, i.e. it actually implements the popular crisp K-means clustering algorithm.

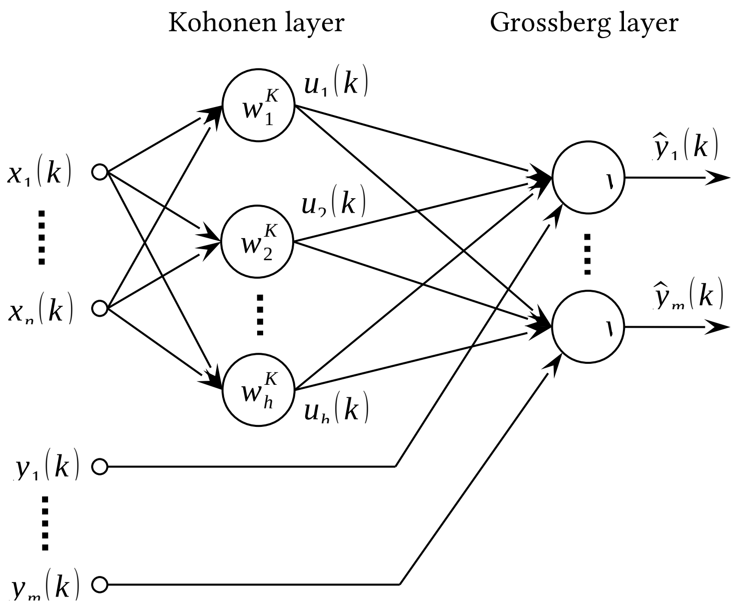

Figure 1: Forward-only counterpropagation network architecture

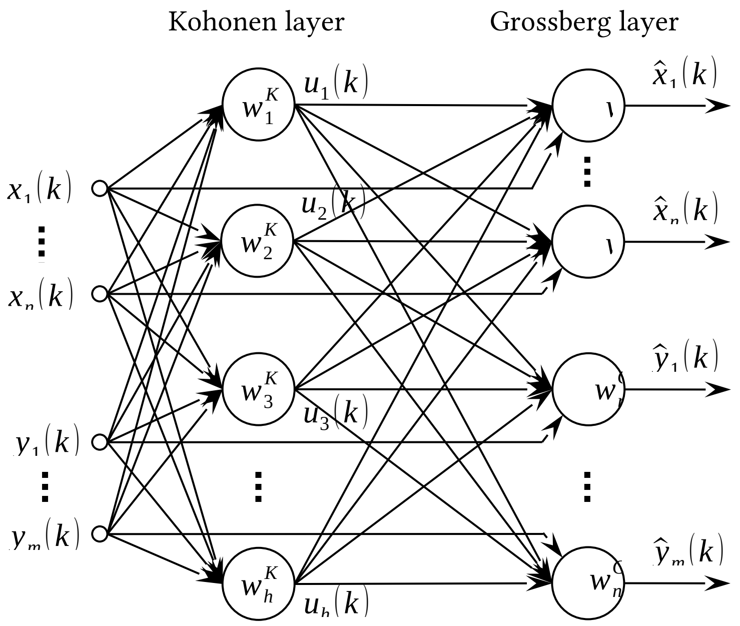

Figure 2: Full counterpropagation network architecture

The output layer is formed by the so-called Grossberg outstars, which are essentially modifications of the standard linear element (Adaline) and implement the mapping

$$\hat{y}(k)=W^G u(k), \tag{3}$$

where $W^G=\{w_{jl}^G\}$ – $(m\times h)$ matrix of synaptic weights tuned in the controlled learning mode.

Outstar neurons of the output (Grossberg) layer are usually trained using a fairly simple algorithm

$$w_{jl}^{G}(k+1)=w_{jl}^{G}(k)+\eta_{G}(k)u_{l}(k)\left(y_{l}(k)-w_{jl}^{G}(k)\right)$$

or using vector notation,

$$w_{j}^{G}(k+1)=w_{j}^{G}(k)+\eta_{G}(k)u^{T}(k)\odot\left(y_{l}(k)E_{h}-w_{l}^{G}(k)\right), \tag{4}$$

where $0<\eta_{G}(k)<1$ is the Grossberg layer learning rate parameter, E_h is a $(1\times h)$ vector of ones, $\odot$ is the element-wise product symbol

So, this neural network acts like a simple lookup table. It gives outputs in steps rather than smoothly, which limits its ability to model complex relationships. Furthermore, using a "Winner Takes All" approach in the Kohonen layer means that during training, only one outstar is adjusted at a time. This makes the overall training process slower.

Given these limitations, modifying the Counterpropagation Neural Network along with its learning methods could enhance its ability to model functions more accurately while speeding up the training process.

3. Neuro-fuzzy counterpropagation network (NFCPN)

The proposed NFCPN maintains the same architecture as traditional CPN networks, but introduces key improvements. Instead of employing a traditional Self-Organizing Map within the Kohonen layer, which traditionally uses a recurrent version of the crisp K-means clustering algorithm, our approach utilizes a modified fuzzy C-means clustering procedure (FCM) [21, 22] in the recurrent form [23], enhancing both learning efficiency and approximation accuracy.

This modification allows for effective handling of situations where classes overlap in feature space, enabling an observation to belong to multiple classes simultaneously. Furthermore, by applying a nonlinear strategy, the network can activate several output layer neurons at once. In contrast, classic CPN networks only trigger one Grossberg outstar during learning, which inherently slows down the process.

Overall, these changes significantly improve the network's performance and adaptability in complex scenarios.

3.1. Kohonen layer learning

To improve the quality and speed of the SOM clustering, we use the so-called "Winner Takes More" (WTM) rule, instead of WTA. This approach utilizes a neighborhood function $\psi(l,g,k)$ that determines the proximity of all other neurons $w_{g}^{K}(k), g=1,2,\dots,l-1,l+1,\dots,h$ to the "winner" $w_{l}^{K}(k)$. For $g=l$, $\psi(l,l,k)=1$, and the value of $\psi(l,g,k)$ decreases with the increase of the distance between vectors $w_{l}^{K}(k)$ and $w_{g}^{K}(k)$.

All centroids – vectors of synaptic weights are tuned according to the modified learning rule

$$w_{l}^{K}(k+1)=w_{l}^{K}(k)+\eta_{K}(k)\psi(l,g,k)\left(x(k)-w_{l}^{K}(k)\right)\ \forall\, l=1,2,\dots,h \tag{5}$$

It is readily seen that (5) is a generalization of the WTA algorithm (2), for which the neighborhood function is a singleton. Unfortunately, there are no formal rules for determining neighborhood functions $\psi(l,g,k)$, hence their selection is based on empirical considerations.

Considering a more practical situation, when each observation can belong to several or all clusters simultaneously, it is beneficial to use a recurrent modification of J.C. Bezdek's FCM algorithm [21] related to optimization of the following objective function

$$J\left(\mu_{l}(k),w_{l}^{K}\right)=\sum_{k=1}^{N}\sum_{l=1}^{h}\mu_{l}^{\beta}(k)\left\|x(k)-w_{l}^{K}\right\|^{2}$$

subject to constraints

$$\sum_{l=1}^{h} \mu_l(k) = 1 \; \forall k = 1,2,\dots,N,$$
$$0 < \sum_{k=1}^{N} \mu_l(k) < N \; \forall l = 1,2,\dots,h.$$

Here $\mu_l(k)$ – degree of fuzzy membership of observation $x(k)$ to l^{th} cluster, $\beta > 0$ – fuzzifier (usually $\beta = 2$), w_l^K – centroid of l^{th} cluster.

Solving the optimization problem based on finding the saddle point of the Lagrange function

$$L\left(\mu_l(k), w_l^K, \lambda(k)\right) = \sum_{k=1}^{N} \sum_{l=1}^{h} \mu_l^{\beta}(k) \left\| x(k) - w_l^K \right\|^2 + \sum_{k=1}^{N} \lambda(k) \left(\sum_{l=1}^{h} \mu_l(k) - 1 \right) \tag{6}$$

(here $\lambda(k)$ – Lagrange multipliers) for $\beta = 2$ leads to the standard FCM algorithm

$$\begin{cases} \text{¿}\, \mu_l(k) = \dfrac{\left\| x(k) - w_l^K \right\|^{-2}}{\sum_{g=1}^{h} \left\| x(k) - w_g^K \right\|^{-2}}, \\ \text{¿}\, w_l^K = \dfrac{\sum_{k=1}^{N} \mu_l^2(k) x(k)}{\sum_{k=1}^{N} \mu_l^2(k)}. \end{cases}$$

To solve the fuzzy clustering problem in online mode, i.e. training the fuzzy Kohonen map, consider a local modification of the Lagrange function (6) in the form [24, 25]

$$L\left(\mu_l(k), w_l^K(k), \lambda(k)\right) = \sum_{l=1}^{h} \mu_l^{\beta}(k) \left\| x(k) - w_l^K(k) \right\|^2 + \lambda(k) \left(\sum_{l=1}^{h} \mu_l(k) - 1 \right).$$

Optimizing it with the K.J. Arrow, L. Hurwitz, H. Uzawa procedure [26], we obtain the following result

$$\begin{cases} \text{¿}\, \mu_l(k) = \dfrac{\left\| x(k) - w_l^K(k) \right\|^{-\beta}}{\sum_{g=1}^{h} \left\| x(k) - w_g^K(k) \right\|^{-\beta}}, \\ \text{¿}\, w_l^K(k+1) = w_l^K(k) + \eta_K(k) \mu_l^{\beta}(k) \left(x(k) - w_l^K(k) \right), \end{cases} \tag{7}$$

which coincides with the D.C. Park, I. Dagher algorithm [27] when $\beta = 2$:

$$\begin{cases} \text{¿}\, \mu_l(k) = \dfrac{\left\| x(k) - w_l^K(k) \right\|^{-2}}{\sum_{g=1}^{h} \left\| x(k) - w_g^K(k) \right\|^{-2}}, \\ \text{¿}\, w_l^K(k+1) = w_l^K(k) + \eta_K(k) \mu_l^2(k) \left(x(k) - w_l^K(k) \right). \end{cases} \tag{8}$$

It is easy to see that (7), (8) structurally coincide with the WTM algorithm (5), but here the neighborhood function is being chosen automatically.

Next, the calculated membership degrees $\mu_l(k) \; \forall l = 1,2,\dots,h$ are fed to the output layer of the network, i.e. vector $u(k)$ is formed not by a single one and a set of zeros, but by membership degrees $\mu_l(k)$, which activate all neurons of the Grossberg output layer.

3.2. Grossberg layer learning

As described above, the Grossberg layer receives the vector $\mu(k)=\left(\mu_1(k),\dots,\mu_l(k),\dots,\mu_h(k)\right)^T$ as input, instead of $u(k)$ in classic CPN. This leads to acceleration of the Grossberg layer learning, because all weights are being updated at each iteration, not only the ones connected to the "winner" of the Kohonen layer. Hence, instead of (2), the learning process of this layer can be rewritten as

$$w_j^G(k+1)=w_j^G(k)+\eta_G(k)\mu^T(k)\oslash\left(y_l(k)E_h-w_l^G(k)\right). \qquad (9)$$

The output layer learning rate parameter $\eta_G(k)$ can be optimized, considering the objective function

$$J\left(w_j^G\right)=\frac{1}{2}\left(y_l(k)-w_l^G\mu(k)\right)^2$$

and its gradient optimization procedure

$$w_j^G(k+1)=w_j^G(k)+\eta_G(k)\left(y_l(k)-w_l^G(k)\mu(k)\right)\mu^T(k). \qquad (10)$$

Optimizing (10) for speed leads to the Kaczmarz-Widrow-Hoff algorithm [28-30] in a form

$$w_j^G(k+1)=w_j^G(k)+\frac{y_l(k)-w_l^G(k)\mu(k)}{\left\|\mu(k)\right\|^2}\mu^T(k). \qquad (11)$$

The balance between filtering and following properties of (11) can be chosen by the following modification [31, 32]:

$$\begin{cases} \text{¿} w_j^G(k+1)=w_j^G(k)+\alpha^{-1}(k)\left(y_l(k)-w_l^G(k)\mu(k)\right)\mu^T(k), \\ \text{¿} \alpha(k)=\gamma\alpha(k-1)+\left\|\mu(k)\right\|^2, 0\le\gamma\le 1, \end{cases} \qquad (12)$$

which coincides with (11) when $\gamma=0$, and becomes a stochastic approximation procedure when $\gamma=1$.

4. Experimental results

4.1. Experimental setup

For the sake of comparison between classic counterpropagation network (CPN) and the proposed neuro-fuzzy counterpropagation network (NFCPN), we use a simple test case with $n=2$ inputs, $h=9$ neurons in the Kohonen layer, and $m=1$ output. Inputs $x_1(k), x_2(k)$ are sampled from the uniform distribution over the interval $[0,1]$, the corresponding output is calculated as $y(k)=\left(x_1(k)^2+x_2(k)^2\right)^{\frac{1}{2}}$.

The first $N=1000$ observations form the training set, another $T=1000$ observations form the test set. Both networks operate in online mode, processing all N training observations sequentially and only once, updating their parameters after each step k. Also, after each step, the mean absolute error (MAE) is calculated over the entire test set, i.e. we monitor how the out-of-sample error changes during the online training process.

First, both networks are trained under the same conditions $\eta_K(k)=\eta_G(k)=0.1 \ \forall k=1,2,\dots,N$, hence we compare WTA principle in CPN versus WTM in NFCPN (Fig. 3). Then, an optimized learning algorithm (12) with various values of parameter γ is used for NFCPN in order to further improve its performance (Fig. 4). Numerical results are presented in Table 1.

Table 1
Effectiveness comparison

Network type and parameters	MAE = 0.1 at $k=¿$	MAE at $k=1000$
CPN, $\eta_K(k)=\eta_G(k)=0.1$	399	0.085
NFCPN, $\eta_K(k)=\eta_G(k)=0.1$	264	0.065
NFCPN, $\eta_K(k)=0.1, \gamma=0$	23	0.07–0.12
NFCPN, $\eta_K(k)=0.1, \gamma=0.9$	118	0.065
NFCPN, $\eta_K(k)=0.1, \gamma=0.99$	695	0.081
NFCPN, $\eta_K(k)=0.1, \gamma=1.0$	–	0.184

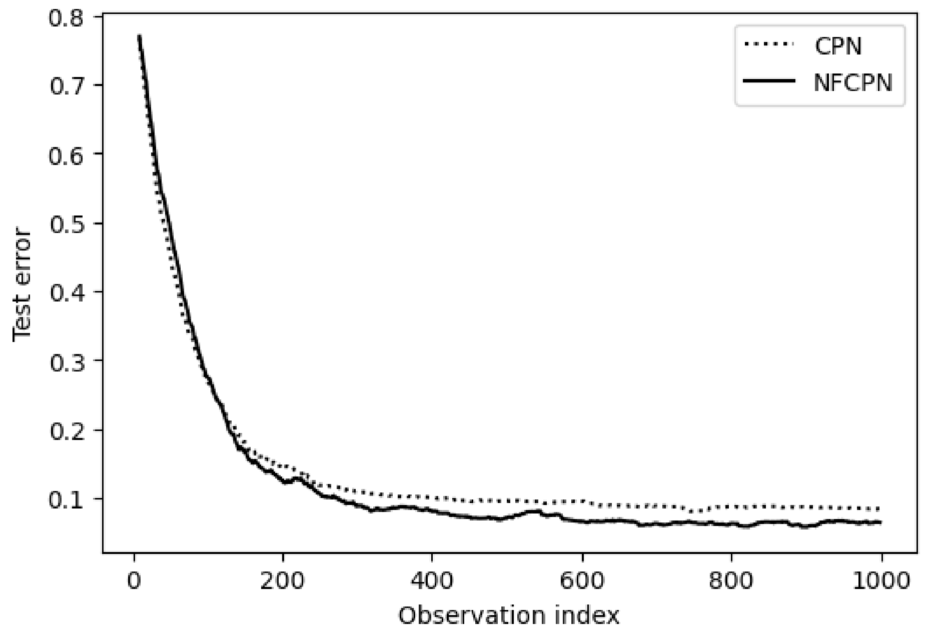

Figure 3: CPN vs NFCPN errors

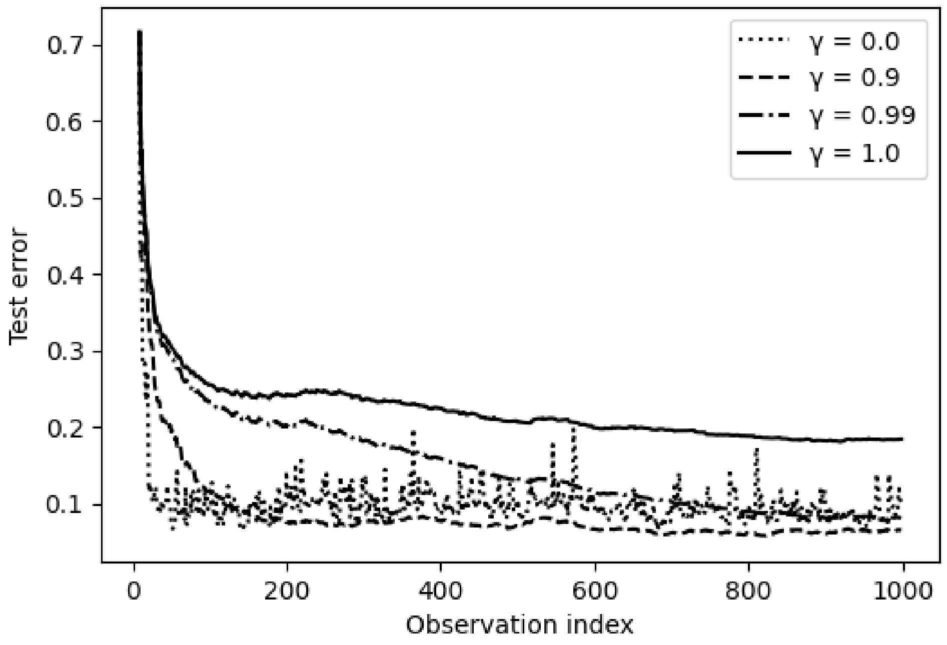

Figure 4: NFCPN errors at different levels of γ

The analysis conducted on the results reveals significant differences between traditional counterpropagation networks (CPN) and neuro-fuzzy counterpropagation networks (NFCPN). These findings highlight the advantages of using NFCPN in achieving faster learning rates and improved accuracy.

4.2. Learning efficiency: MAE comparison across networks

Mean Absolute Error (MAE), a key metric for evaluating model performance, was calculated at various stages of training. The results demonstrate that:

- Classic CPN: At $k=399$ iterations, the classic CPN achieved MAE level of 0.1.
- NFCPN: In comparison, the neuro-fuzzy counterpart reached a similar MAE level of 0.1 at $k=264$ iterations.

This indicates that NFCPN requires fewer training cycles to achieve comparable accuracy, suggesting superior learning efficiency compared to classic CPN. Further results reinforce this conclusion.

At $k=1000$:

- Classic CPN: The MAE stabilized at 0.085.
- NFCPN: Achieved an improved MAE of 0.065, showcasing greater accuracy even as training progressed.

These results collectively demonstrate that, with the same training parameters, NFCPN learns approximately 1.5 times faster than CPN and is by 24% more accurate in performing the given task.

4.3. Adjusting learning dynamics: the role of gamma parameter (γ)

The study also explored different configurations for enhancing learning performance using algorithm (12) and adjusting the gamma parameter (γ).

Initially, γ was set to 0. This setting significantly increased the learning rate by over 10 times in comparison to the classical learning algorithm with $\eta_K(k)=\eta_G(k)=0.1$. However, this improvement came at a cost – training became noisy, lacking effective filtering properties. To address this trade-off and improve the filtering characteristics of the algorithm without compromising learning speed, gamma was gradually increased. With $\gamma=0.9$, errors comparable to those achieved with a fixed $\eta_G(k)=0.1$ were observed. Additionally, this configuration maintained an impressive learning rate that was about 2.2 times faster than with the classical learning algorithm.

This experimentation underscores the importance of fine-tuning gamma to achieve a balance between noise reduction and efficient learning rates. By carefully controlling gamma, it is possible to optimize both filtering properties and following characteristics (i.e., adaptability to changes in non-stationary data streams).

4.4. Key findings summary

- Learning rate: NFCPN consistently outperforms CPN by achieving comparable or better MAE with fewer training iterations.
- Accuracy enhancement: The improved performance of NFCPN results in a 24% increase in accuracy over classic CPN under the same conditions.
- Parameter optimization: Modifying gamma allows for precise control over learning dynamics, balancing between noisy and stable training processes. Adjusting gamma to higher values enhances filtering properties without significantly compromising on learning speed.

5. Conclusions

We have introduced a fuzzy modification to a counterpropagation network, enhancing its ability to handle situations where data categories overlap. This means an item can belong to multiple classes simultaneously, which is common in real-world scenarios.

Our modifications improve the network's learning efficiency and enable it to address a wider range of problems in real-time data processing. Additionally, this enhanced version is simpler mathematically and requires less training data compared to traditional methods. It also adapts smoothly as new, varied data arrives, which is crucial for handling dynamic information streams.

Experiments demonstrate that our modified network performs effectively and outperforms the standard CPN model. These results suggest that neuro-fuzzy counterpropagation networks hold significant potential in real-time data processing tasks where both efficiency and accuracy are critical. The ability to adjust gamma parameter offers flexibility, enabling the network to adapt to varying levels of non-stationarity in input data streams.

Further research could explore additional parameter configurations or investigate how the proposed approach generalizes across different tasks. We also aim to explore different clustering techniques within the hidden layer of NFCPN to further enhance its capabilities. Such advancements would likely enhance the applicability of counterpropagation networks across a broader range of real-world scenarios.

Declaration on Generative AI

During the preparation of this work, the authors used Grammarly in order to: Grammar and spelling check. After using this tool, the authors reviewed and edited the content as needed and take full responsibility for the publication's content.

References

[1] R. Rojas, Neural Networks. A Systematic Introduction, Springer-Verlag Berlin, 1996.

[2] L. H. Tsoukalas, R. E. Uhrig, Fuzzy and Neural Approaches in Engineering, John Wiley and Sons, 1997.

[3] R. Hecht-Nielsen, Counter-propagation Network, in: Proceedings of IEEE First International Conference on Neural Networks, volume 2, 1987, pp. 19-32.

[4] R. Hecht-Nielsen. "Counterpropagation networks." Applied optics 26 (1987): 4979-4984.

[5] R. Hecht-Nielsen. "Applications of counterpropagation networks." Neural networks 1 (1988): 131-139.

[6] B. Bajželj, V. Drgan "Hepatotoxicity Modeling Using Counter-Propagation Artificial Neural Networks: Handling an Imbalanced Classification Problem". Molecules 25 (2020): 481.

[7] R. Rahmat, Y. Harahap, D. Rachmawati "Counter-propagation Neural Network for Brain Tumor Classification" J. Phys.: Conf. Ser. 1566 (2020) 012128.

[8] S. Sutha, N. Gnanambigai, P. Dinadayalan "Extended Self-Organizing Map With Ubiquitous Counter Propagation Network In Classification For Diabetic Database" Solid State Technology 63 (2020)

[9] N. Vivekanandan, K. Rajeswari, S. Salve, N. Kanna, Precision Smoke Detection System with Counter Propagation Neural Network and Electronic Olfactory. In: De, A., Mukherjee, P.P., Pati, S., Biswas, A. (eds) Recent Trends in Mechanical Engineering. ICRAME 2024. Lecture Notes in Mechanical Engineering. Springer, Singapore, pp. 235–247.

[10] B. Pratap "Analysis of mechanical properties of fly ash and bauxite residue based geopolymer concrete using ANN, Random Forest and Counter propagation neural network." Asian J Civ Eng 25 (2024): 4303–4317.

[11] R. Mutra, J. Srinivas, R. Rządkowski "An optimal parameter identification approach in foil bearing supported high-speed turbocharger rotor system" Arch Appl Mech 91 (2021): 1557–1575.

[12] A. Kaveh, Structural Optimization by Gradient-Based Neural Networks, In: Applications of Artificial Neural Networks and Machine Learning in Civil Engineering. Studies in Computational Intelligence, Vol 1168, 2024, Springer, Cham, pp 147–163.

[13] G. Kayhan , İ. İşeri "Counter Propagation Network Based Extreme Learning Machine." Neural Process Lett 55 (2023): 857–872.

[14] A. Taheri "A New Robust Digital Image Watermarking Technique Using Relations between Wavelet Coefficients Transform and Full Counter Propagation Neural Network" International Journal of Recent Research in Electrical and Electronics Engineering (IJRREEE) 7 (2020): 1–12.

[15] D-J. Jwo, A. Biswal, I. Mir "Artificial Neural Networks for Navigation Systems: A Review of Recent Research." Applied Sciences 13 (2023):4475.

[16] J. Kacprzyk, W. Pedrycz (Eds.) Springer Handbook of Computational Intelligence, Springer, Berlin-Heidelberg, 2015.

[17] T. Yamakawa, E. Uchino, T. Miki, H. Kusanagi, A neo fuzzy neuron and its applications to system identification and prediction of the system behavior, in: Proceedings of 2nd Int. Conf. on Fuzzy Logic and Neural Networks, 1992, pp. 477-483.

[18] T. Miki, T. Yamakawa, Analog implementation of neo-fuzzy neuron and its on-board learning, in: Computational Intelligence and Applications, WSES Press, Piraeus, 1999, pp. 144-149.

[19] T. Kohonen, Self-Organizing Maps, Springer-Verlag, Berlin, 1995.

[20] T. Kohonen, Improved versions of learning vector quantization, in: Proceedings of 1990 IJCNN International Joint Conference on Neural Networks, volume 1, San Diego, CA, USA, 1990, pp. 545-550.

[21] J. C. Bezdek. Pattern Recognition with Fuzzy Objective Function Algorithms, Plenum Press, New York, 1981.

[22] F. Hoeppner, F. Klawonn, R. Kruse, T. Runkler, Fuzzy cluster analysis and image recognition, Wiley & Sons, Chichester, 1999.

[23] Ye. Bodyanskiy, Computational intelligence techniques for data analysis, in: Lecture Notes in Informatics, volume 72, GI, Bonn, 2005, pp. 15-36.

[24] Ye. Bodyanskiy, Ye. Gorshkov, V. Kolodyazhniy, New recursive learning algorithms for fuzzy Kohonen clustering network, in: Proceedings of 17th Int. Workshop on Nonlinear-Dynamics of Electronic Systems. Rapperswil, Switzerland, June 21-24, 2009, pp. 58-61.

[25] Ye. Bodyanskiy, A. Deineko, F. Eze "Kernel Fuzzy Kohonen's Clustering Neural Network and It's Recursive Learning." Automatic Control and Computer Sciences 52 (2018): 166-174.

[26] K. J. Arrow, L. Hurwicz, H. Uzawa. Studies in linear and non-linear programming, Stanford University Press, 1958.

[27] D. C. Park, I. Dagher, Gradient based fuzzy c-means (GBFCM) algorithm, in: Proceedings of 1994 IEEE International Conference on Neural Networks (ICNN'94), volume 3, Orlando, FL, USA, 1994, pp. 1626-1631

[28] S. Kaczmarz "Angenäherte Auflösung von Systemen linearer Gleichungen." Bulletin International de l'Académie Polonaise des Sciences et des Lettres. Classe des Sciences Mathématiques et Naturelles. Série A, Sciences Mathématiques 35 (1937): 355–357.

[29] S. Kaczmarz "Approximate solution of systems of linear equations." International Journal of Control 57 (1993): 1269-1271.

[30] B. Widrow, M. Hoff, Adaptive Switching Circuits, in: 1960 IRE WESCON Convention Record, part 4, 1960, pp. 96-104.

[31] Ye. Bodyanskiy, O. Chala, I. Izonin, S. Popov Simple neuro-fuzzy system with combined learning for pattern recognition under conditions of short training set in medical diagnostics tasks, in: Proceedings of the 5th International Conference on Informatics & Data-Driven Medicine (IDDM 2022), Lyon, France, November 18-20, 2022, pp. 1-8.

[32] Ye. Bodyanskiy, I. Kokshenev, V. Kolodyazhniy, An adaptive learning algorithm for a neo fuzzy neuron, in: Proceedings of the 3rd Int. Conf. of European Union Society for Fuzzy Logic and Technology (EUSFLAT 2003), Zittau, Germany, 10-12 September, 2003, pp. 375-379.