Preprint Version

Publisher: Springer, Nature

Collected work provisionally entitled: Applications of Fuzzy Techniques: Proceedings of the 2022 Annual Conference of the North American Fuzzy Information Processing Society NAFIPS

Book series: Advances in Intelligent Systems and Computing

Edited by: Scott Dick, Vladik Kreinovich, Pawan Lingras

Multiple Hidden Layered CEFYDRA: Cluster-first Explainable FuzzY-based Deep self-Reorganizing Algorithm

Javier Viaña[1][0000-0002-0563-784X], Stephan Ralescu[2][0000-0002-3969-1342], Vladik Kreinovich[3][0000-0002-1244-1650], Anca Ralescu[4][0000-0002-7564-3540], and Kelly Cohen[5][0000-0002-8655-1465]

[1,2,3,4,5] University of Cincinnati, Cincinnati OH 45219, USA
[1] Massachusetts Institute of Technology, Cambridge MA 02139, USA
vianajr@mail.uc.edu
vianajr@mit.edu

Abstract. We propose a deep learning algorithm that breaks with the paradigm of weights and activation functions, CEFYDRA, a network of cluster-first fuzzy-based regression algorithms. In this paper, we cover the generalization of CEFYDRA to multilayered deep architectures. First, we provide the three laws that make this generalization possible: Displacement, Substitution and Composition. Then, we obtain the update formulas for the parameters of deep hidden layers. Finally, we show the pseudocode for prediction and update. We also briefly mention the reasons to believe that this algorithm, named CEFYDRA, is explainable and has the ability for plastic reorganization.

Keywords: Explainable AI, Neural Networks, Fuzzy Logic, Gradient Descent, Deep Learning.

1 Introduction

In the last year, multiple explainable neural network algorithms were created, e.g., [1-4]. Some of which have proved to be very successful in a variety of fields, such as the discovery of exoplanets [5] or industrial automation [6]. In fact, based on the number of publications and patents published over the last decade, it can be said that the neural networks have captured the attention of most of the research carried out in computer science. Despite the advances made, a new core methodology different from the classic mechanism of weights and activation functions is rarely proposed [7-8]. To a certain extent this implies a limitation in the type of problems that we can tackle.

The research on alternative and novel basic neural architectures should not decrease. Even so, the main techniques to improve the performance in a problem solved with deep learning continue to be the iterative (trial and error) modification of the number of layers, units, or the shape of the activation functions. Meanwhile, the avenue of varying the inner functioning of the neural unit itself remains partially unexploited.

2 On the Learning Formulas for the Hidden Layer

One of the goals of this research is to provide an alternative to the standard of weights, biases, and activation functions of a regular neural network. We proposed a model called CEFYDRA, an architecture whose units are based on the algorithms covered in [9-13]. At every unit of the network, the inputs are grouped in fuzzy clusters (whose degree of membership is defined through Cauchy membership functions), and mapped with multidimensional logistic functions. The output of a unit is obtained by merging the predictions of all the clusters using a Takagi-Sugeno-Kang approach. All the parameters are optimized following a gradient descent learning.

In this section we focus on how to update the parameters of deep hidden layers of a CEFYDRA.

In our formulation, the upper left index in parenthesis represents the layer reference (0 for output, 1 for the first hidden starting from the right), and the bottom index represents the unit within that layer. We use $\mathbf{x_i}, \mathbf{y_i}$, and $\hat{\mathbf{y}}_\mathbf{i}$ to denote the i^{th} input, output, and predicted output, respectively. The symbols t and w represent the output and a generic parameter of a given unit. Finally, indexes k and j refer to the cluster and input dimension within the unit. The membership functions are obtained from Cauchy distributions,

$$ {}^{(\lambda)}_{h_\lambda}\mu_c(\mathbf{x_i}) = \frac{1}{1 + \left\| {}^{(\lambda)}_{\boldsymbol{h_\lambda}}\mathbf{u_c} \cdot {}^{(\lambda+1)}\mathbf{t}(\mathbf{x_i}) + {}^{(\lambda)}_{\boldsymbol{h_\lambda}}\mathbf{v_c} \right\|^2} = \left[1 + \sum_{h=1}^{H} \left({}^{(\lambda)}_{h_\lambda}u_{ch} \cdot {}^{(\lambda+1)}t_h(\mathbf{x_i}) + {}^{(\lambda)}_{h_\lambda}v_{ch} \right)^2 \right]^{-1}. \tag{1} $$

In [7-8], the functions for the approximation of each cluster are linear, however, we want to constrain the output of each unit between 0 and 1. Thus, we use logistic functions instead, for a generic unit h_λ within layer λ,

$$ {}^{(\lambda)}_{h_\lambda}r_c(\mathbf{x_i}) = \frac{1}{1 + \exp\left(- {}^{(\lambda)}_{\boldsymbol{h_\lambda}}\mathbf{m_c} \cdot {}^{(\lambda+1)}\mathbf{t}(\mathbf{x_i}) - {}^{(\lambda)}_{h_\lambda}n_c \right)}. \tag{2} $$

Therefore, the parameters of each cluster are ${}^{(\lambda)}_{\boldsymbol{h_\lambda}}\mathbf{u_c}$, ${}^{(\lambda)}_{\boldsymbol{h_\lambda}}\mathbf{v_c}$, ${}^{(\lambda)}_{\boldsymbol{h_\lambda}}\mathbf{m_c}$ and ${}^{(\lambda)}_{h_\lambda}n_c$.

With that set up, we start from the premise that between any output of a unit from the output layer and any other output of a unit from the first hidden layer exists a linear proportionality,

$$ \frac{\partial \hat{y}_l(\mathbf{x_i})}{\partial {}^{(1)}_{h_1}w_{kj}} = K \frac{\partial {}^{(1)}t_{h_1}(\mathbf{x_i})}{\partial {}^{(1)}_{h_1}w_{kj}}, \tag{3} $$

where K is a parameter that depends on the connection between these two units. It can be demonstrated that

$$ \begin{aligned} K = \begin{bmatrix} {}^{(1)} & {}^{(0)} \\ {}_{h_1} & {}_{l} \end{bmatrix} \phi(\mathbf{x_i}) \stackrel{\text{def}}{=} & \left[\sum_{c=1}^{C} {}^{(0)}_{l}\mu_c(\mathbf{t_i}) \right]^{-1} \sum_{c=1}^{C} \Big\{ {}^{(0)}_{l}\mu_c(\mathbf{t_i}) \Big\{ -2 {}^{(0)}_{l}\mu_c(\mathbf{t_i}) \\ & \cdot \left({}^{(0)}_{l}u_{ch} \cdot {}^{(1)}t_{h_1}(\mathbf{x_i}) + {}^{(0)}_{l}v_{ch} \right) \cdot {}^{(0)}_{l}u_{ch} \cdot \left[{}^{(0)}_{l}r_c(\mathbf{t_i}) - \hat{y}_l(\mathbf{x_i}) \right] \\ & + {}^{(0)}_{l}m_{ch} \cdot {}^{(0)}_{l}\varpi_c(\mathbf{t_i}) \Big\} \Big\}. \end{aligned} \tag{4} $$

where ${}^{(0)}_{l}\varpi_c(\mathbf{t_i}) \stackrel{\text{def}}{=} \left[{}^{(0)}_{l}r_c(\mathbf{t_i})\right]^2 \cdot \exp\left({}^{(0)}_{l}s_c(\mathbf{t_i})\right)$.

Due to this decoupling in the consecutive outputs, we can expect a generalized formula that relates any two non-consecutive outputs (from any two layers) of the type

$$\frac{\partial \hat{y}_l(\mathbf{x_i})}{\partial\, {}^{(\lambda)}_{h_\lambda}w_{kj}} = \frac{\partial\, {}^{(\lambda)}t_{h_\lambda}(\mathbf{x_i})}{\partial\, {}^{(\lambda)}_{h_\lambda}w_{kj}} \prod \phi(\mathbf{x_i}). \tag{5}$$

We use the upper left index to distinguish between the hidden layers (Fig. 1). For a given layer λ, unit h_λ, the output variable is ${}^{(\lambda)}t_{h_\lambda}$.

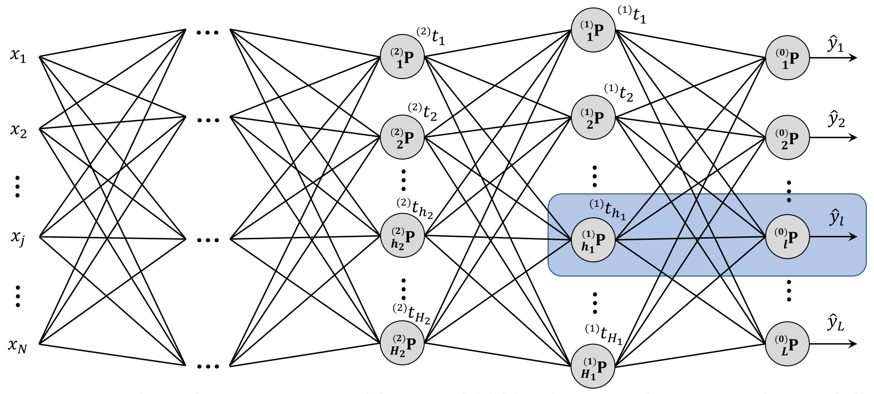


Fig. 1. Representation of a CEFYDRA with several hidden layers and an output layer of dimension L. The two units highlighted referred to equation (1).

There are three laws that we can derive from (1):

Displacement Law. We can consider the equation (1) between every two consecutive layers. If we use the hidden layers 2 and 1, we get

$$\frac{\partial\, {}^{(1)}t_{h_1}(\mathbf{x_i})}{\partial\, {}^{(2)}_{h_2}w_{kj}} = \begin{bmatrix}(2)(1)\\ h_2\ h_1\end{bmatrix}\phi(\mathbf{x_i})\frac{\partial\, {}^{(2)}t_{h_2}(\mathbf{x_i})}{\partial\, {}^{(2)}_{h_2}w_{kj}}, \tag{6}$$

where

$$\begin{aligned}\begin{bmatrix}(2)(1)\\ h_2\ h_1\end{bmatrix}\phi(\mathbf{x_i}) \stackrel{\text{def}}{=} &\left[\sum_{c=1}^{C} {}^{(1)}_{h_1}\mu_c\left({}^{(2)}\mathbf{t_i}\right)\right]^{-1} \sum_{c=1}^{C}\Big\{ {}^{(1)}_{h_1}\mu_c\left({}^{(2)}\mathbf{t_i}\right) \\ &\cdot\Big\{-2\,{}^{(1)}_{h_1}\mu_c\left({}^{(2)}\mathbf{t_i}\right)\cdot\left[{}^{(1)}_{h_1}u_{ch_2}\cdot {}^{(2)}t_{h_2}(\mathbf{x_i}) + {}^{(1)}_{h_1}v_{ch_2}\right]\cdot {}^{(1)}_{h_1}u_{ch_2} \\ &\cdot\left[{}^{(1)}_{h_1}r_c\left({}^{(2)}\mathbf{t_i}\right) - {}^{(1)}t_{h_1}(\mathbf{x_i})\right] + {}^{(1)}_{h_1}m_{ch_2}\cdot {}^{(1)}_{h_1}\varpi_c\left({}^{(2)}\mathbf{t_i}\right)\Big\}\Big\}.\end{aligned} \tag{7}$$

To obtain $\begin{bmatrix}(2)(1)\\ h_2\ h_1\end{bmatrix}\phi(\mathbf{x_i})$, we treat ${}^{(1)}t_{h_1}$ as the output $\hat{y}_l$ we used in (1) and ${}^{(2)}\mathbf{t_i}$ instead of the input $\mathbf{t_i}$. Similarly, we also adapt the parameters of $\begin{bmatrix}(2)(1)\\ h_2\ h_1\end{bmatrix}\phi(\mathbf{x_i})$ to those of the h_1^{th} unit of the first hidden layer. Equation (6) is the particular form of the Displacement Law between the unit h_1 of layer 1 and the unit h_2 of layer 2 (Fig. 2).

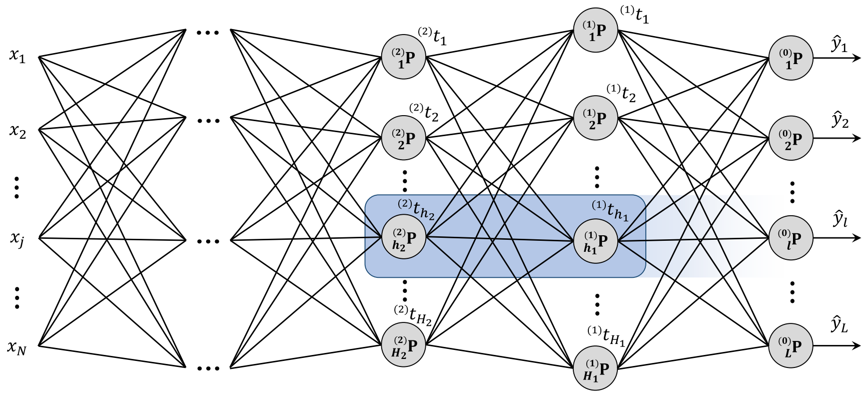


Fig. 2. Highlight of the two units within the CEFYDRA that refer to equation (6), i.e., the Displacement Law in its particular form between hidden layers 1 and 2.

We now translate the particular form of the Displacement Law (6) to its general form by considering the hidden layers λ and, $\lambda - 1$. We focus on the α^{th} output of layer λ (${}^{(\lambda)}t_\alpha$) and the β^{th} output of layer $\lambda - 1$ (${}^{(\lambda-1)}t_\beta$) as show in Fig. 3.

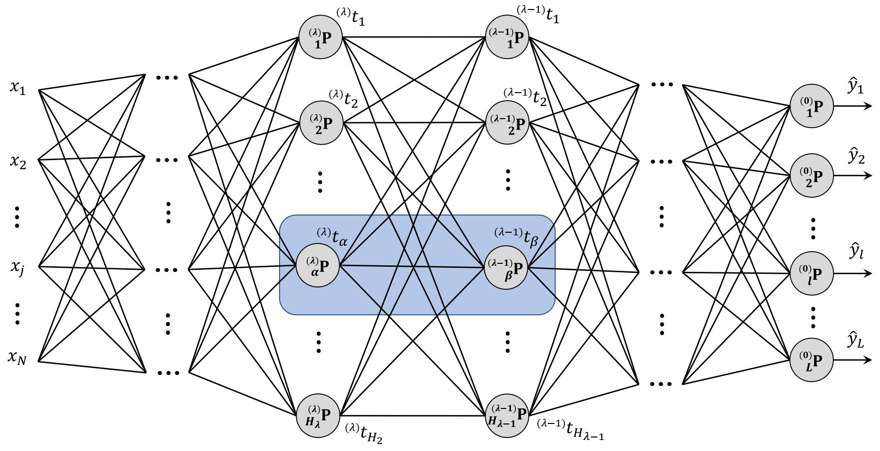


Fig. 3. Highlight of the two units within the CEFYDRA network that refer to equation (8), i.e., the Displacement Law in its general form between hidden layers $\lambda - 1$ and λ.

The resulting general form is

$$\frac{\partial\, {}^{(\lambda-1)}t_\beta(\mathbf{x_i})}{\partial\, {}^{(\lambda)}_{\alpha}w_{kj}} = \begin{bmatrix} (\lambda) & (\lambda-1) \\ \alpha & \beta \end{bmatrix} \phi(\mathbf{x_i}) \frac{\partial\, {}^{(\lambda)}t_\alpha(\mathbf{x_i})}{\partial\, {}^{(\lambda)}_{\alpha}w_{kj}}. \tag{8}$$

Substitution Law. This Law comes from the application of the chain rule in the derivatives of (1) to consider any other parameter of the system,

$$\frac{\partial \hat{y}_l(\mathbf{x_i})}{\partial p} \frac{\partial p}{\partial\, {}^{(1)}_{h_1}w_{kj}} = \begin{bmatrix} (1) & (0) \\ h_1 & l \end{bmatrix} \phi(\mathbf{x_i}) \frac{\partial\, {}^{(1)}t_{h_1}(\mathbf{x_i})}{\partial p} \frac{\partial p}{\partial\, {}^{(1)}_{h_1}w_{kj}}, \tag{9}$$

$$\frac{\partial \hat{y}_l(\mathbf{x_i})}{\partial p} = \begin{bmatrix} (1)(0) \\ h_1 \ \ l \end{bmatrix} \phi(\mathbf{x_i}) \frac{\partial\, {}^{(1)}t_{h_1}(\mathbf{x_i})}{\partial p}. \tag{10}$$

If we consider the parameter $p = {}^{(2)}_{h_2}w_{kj}$ for the layer 2 in formula (10), then

$$\frac{\partial \hat{y}_l(\mathbf{x_i})}{\partial\, {}^{(2)}_{h_2}w_{kj}} = \begin{bmatrix} (1)(0) \\ h_1 \ \ l \end{bmatrix} \phi(\mathbf{x_i}) \frac{\partial\, {}^{(1)}t_{h_1}(\mathbf{x_i})}{\partial\, {}^{(2)}_{h_2}w_{kj}}. \tag{11}$$

In the general form, we can take the general expression (8) obtained for the Displacement Law and consider any layer π and any unit γ within the layer π,

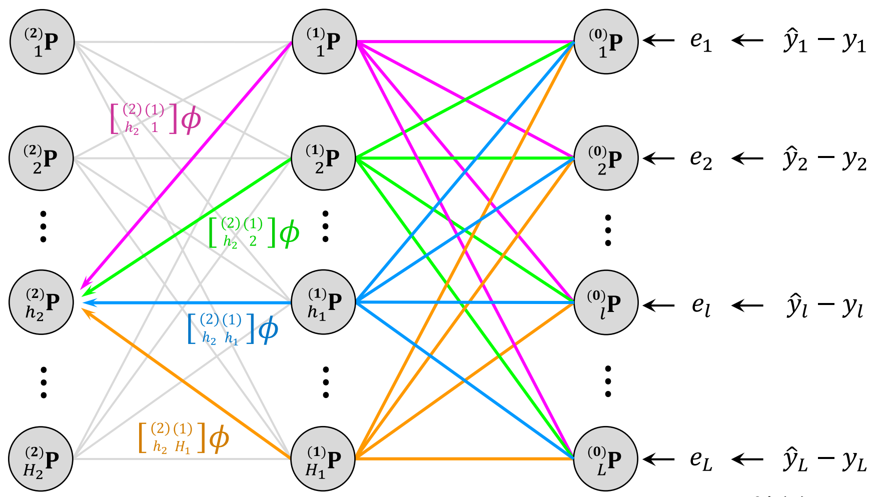
$$\frac{\partial\, {}^{(\lambda-1)}t_{\beta}(\mathbf{x_i})}{\partial\, {}^{(\pi)}_{\gamma}w_{kj}} = \begin{bmatrix} (\lambda)(\lambda-1) \\ \alpha \ \ \ \beta \end{bmatrix} \phi(\mathbf{x_i}) \frac{\partial\, {}^{(\lambda)}t_{\alpha}(\mathbf{x_i})}{\partial\, {}^{(\pi)}_{\gamma}w_{kj}}. \tag{12}$$

Composition Law. The Composition Law is achieved from the combination of both the Displacement and Substitution Laws. For the particular cases (6) and (10),

$$\frac{\partial \hat{y}_l(\mathbf{x_i})}{\partial\, {}^{(2)}_{h_2}w_{kj}} = \begin{bmatrix} (2)(1) \\ h_2 \ h_1 \end{bmatrix} \phi(\mathbf{x_i}) \cdot \begin{bmatrix} (1)(0) \\ h_1 \ \ l \end{bmatrix} \phi(\mathbf{x_i}) \frac{\partial\, {}^{(2)}t_{h_2}(\mathbf{x_i})}{\partial\, {}^{(2)}_{h_2}w_{kj}}. \tag{13}$$

Equation (13) is certainly interesting, as it shows that $\frac{\partial \hat{y}_l(\mathbf{x_i})}{\partial\, {}^{(2)}_{h_2}w_{kj}}$ is not only dependent on $\begin{bmatrix} (1)(0) \\ h_1 \ \ l \end{bmatrix} \phi(\mathbf{x_i})$ and $\frac{\partial\, {}^{(2)}t_{h_2}(\mathbf{x_i})}{\partial\, {}^{(2)}_{h_2}w_{kj}}$ (which can be understood since in $\frac{\partial \hat{y}_l(\mathbf{x_i})}{\partial\, {}^{(2)}_{h_2}w_{kj}}$ we have both the information of the unit h_2 of the second hidden layer and the information of the unit l of the output layer) but it is also dependent on $\begin{bmatrix} (2)(1) \\ h_2 \ h_1 \end{bmatrix} \phi(\mathbf{x_i})$, i.e., the information of the unit h_1 of the first hidden layer. This might seem confusing, since h_1 is not present in either $\frac{\partial \hat{y}_l(\mathbf{x_i})}{\partial\, {}^{(2)}_{h_2}w_{kj}}$ nor $\frac{\partial\, {}^{(2)}t_{h_2}(\mathbf{x_i})}{\partial\, {}^{(2)}_{h_2}w_{kj}}$. In fact, there is no specification for h_1, therefore h_1 could be any unit of the first hidden layer. This comes from the fact that the system is overdetermined due to the fully connected network. In other words, there are many paths that connect both h_2 and l, and they all provide a different result of $\frac{\partial \hat{y}_l(\mathbf{x_i})}{\partial\, {}^{(2)}_{h_2}w_{kj}}$. Thus, we suggest that $\frac{\partial \hat{y}_l(\mathbf{x_i})}{\partial\, {}^{(2)}_{h_2}w_{kj}}$ should be calculated by averaging the value obtained from all the different paths (Fig. 4),

$$\frac{\partial \hat{y}_l(\mathbf{x_i})}{\partial\, {}^{(2)}_{h_2}w_{kj}} = \frac{1}{H_1} \sum_{h_1=1}^{H_1} \left[\begin{bmatrix} (2)(1) \\ h_2 \ h_1 \end{bmatrix} \phi(\mathbf{x_i}) \cdot \begin{bmatrix} (1)(0) \\ h_1 \ \ l \end{bmatrix} \phi(\mathbf{x_i}) \frac{\partial\, {}^{(2)}t_{h_2}(\mathbf{x_i})}{\partial\, {}^{(2)}_{h_2}w_{kj}} \right]. \tag{14}$$



${}^{(2)}_{1}\mathbf{P}$
${}^{(2)}_{2}\mathbf{P}$
${}^{(2)}_{h_2}\mathbf{P}$
${}^{(2)}_{H_2}\mathbf{P}$
${}^{(1)}_{1}\mathbf{P}$
${}^{(1)}_{2}\mathbf{P}$
${}^{(1)}_{h_1}\mathbf{P}$
${}^{(1)}_{H_1}\mathbf{P}$
${}^{(0)}_{1}\mathbf{P}$
${}^{(0)}_{2}\mathbf{P}$
${}^{(0)}_{l}\mathbf{P}$
${}^{(0)}_{L}\mathbf{P}$
$\left[\begin{smallmatrix}(2)&(1)\\h_2&1\end{smallmatrix}\right]\phi$
$\left[\begin{smallmatrix}(2)&(1)\\h_2&2\end{smallmatrix}\right]\phi$
$\left[\begin{smallmatrix}(2)&(1)\\h_2&h_1\end{smallmatrix}\right]\phi$
$\left[\begin{smallmatrix}(2)&(1)\\h_2&H_1\end{smallmatrix}\right]\phi$
$\leftarrow e_1 \leftarrow \hat{y}_1 - y_1$
$\leftarrow e_2 \leftarrow \hat{y}_2 - y_2$
$\leftarrow e_l \leftarrow \hat{y}_l - y_l$
$\leftarrow e_L \leftarrow \hat{y}_L - y_L$

Fig. 4. Graphical representation of all the possible paths in the calculation of $\frac{\partial \hat{y}_l(\mathbf{x_i})}{\partial\, {}^{(2)}_{h_2}w_{kj}}$. The term e_{il} represents the error of each output unit $y_{il} - \hat{y}_l(\mathbf{x_i})$.

To calculate the general form of (13), we can now study the expression of $\frac{\partial \hat{y}_l(\mathbf{x_i})}{\partial\, {}^{(3)}_{h_3}w_{kj}}$. We start from the expression of the particular Composition Law, and we apply the Substitution Law, i.e., instead of deriving with respect to ${}^{(2)}_{h_2}w_{kj}$, we derive with respect to ${}^{(3)}_{h_3}w_{kj}$, where 3 is the third hidden layer and h_3 is a generic hidden neuron, again, the factors ${}^{(0)}_{l}\phi(\mathbf{x_i})$ and ${}^{(1)}_{h_1}\phi(\mathbf{x_i})$ do not depend on ${}^{(2)}_{h_2}w_{kj}$ or ${}^{(3)}_{h_3}w_{kj}$, thus

$$\frac{\partial \hat{y}_l(\mathbf{x_i})}{\partial\, {}^{(3)}_{h_3}w_{kj}} = \left[\begin{smallmatrix}(2)&(1)\\h_2&h_1\end{smallmatrix}\right]\phi(\mathbf{x_i}) \cdot \left[\begin{smallmatrix}(1)&(0)\\h_1&l\end{smallmatrix}\right]\phi(\mathbf{x_i})\, \frac{\partial\, {}^{(2)}t_{h_2}(\mathbf{x_i})}{\partial\, {}^{(3)}_{h_3}w_{kj}}, \tag{15}$$

now we consider the Displacement Law for layers 3 and 2, let us call ${}^{(3)}t_{h_3}$ the input before ${}^{(2)}t_{h_2}$,

$$\frac{\partial\, {}^{(2)}t_{h_2}(\mathbf{x_i})}{\partial\, {}^{(3)}_{h_3}w_{kj}} = \left[\begin{smallmatrix}(3)&(2)\\h_3&h_2\end{smallmatrix}\right]\phi(\mathbf{x_i})\, \frac{\partial\, {}^{(3)}t_{h_3}(\mathbf{x_i})}{\partial\, {}^{(3)}_{h_3}w_{kj}}, \tag{16}$$

and we combine it with the result obtained in (15),

$$\frac{\partial \hat{y}_l(\mathbf{x_i})}{\partial\, {}^{(3)}_{h_3}w_{kj}} = \left[\begin{smallmatrix}(3)&(2)\\h_3&h_2\end{smallmatrix}\right]\phi(\mathbf{x_i}) \cdot \left[\begin{smallmatrix}(2)&(1)\\h_2&h_1\end{smallmatrix}\right]\phi(\mathbf{x_i}) \cdot \left[\begin{smallmatrix}(1)&(0)\\h_1&l\end{smallmatrix}\right]\phi(\mathbf{x_i})\, \frac{\partial\, {}^{(3)}t_{h_3}(\mathbf{x_i})}{\partial\, {}^{(3)}_{h_3}w_{kj}}. \tag{17}$$

As it happened in (13), this particular case of the Composition Law (17), which defines the learning of the third layer's parameters, also depends on the path and the connections chosen in layers 1 and 2. Hence we use the average of all the possible paths as shown in Fig. 5,

$$\frac{\partial \hat{y}_l(\mathbf{x_i})}{\partial\, {}^{(3)}_{h_3}w_{kj}} = \frac{1}{H_1 \cdot H_2} \sum_{h_2=1}^{H_2} \sum_{h_1=1}^{H_1} \left\{ \left[\begin{smallmatrix}(3)&(2)\\h_3&h_2\end{smallmatrix}\right]\phi(\mathbf{x_i}) \cdot \left[\begin{smallmatrix}(2)&(1)\\h_2&h_1\end{smallmatrix}\right]\phi(\mathbf{x_i}) \cdot \left[\begin{smallmatrix}(1)&(0)\\h_1&l\end{smallmatrix}\right]\phi(\mathbf{x_i})\, \frac{\partial\, {}^{(3)}t_{h_3}(\mathbf{x_i})}{\partial\, {}^{(3)}_{h_3}w_{kj}} \right\}, \tag{18}$$

$$\frac{\partial \hat{y}_l(\mathbf{x_i})}{\partial\, {}^{(3)}_{h_3}w_{kj}} = \frac{\partial\, {}^{(3)}t_{h_3}(\mathbf{x_i})}{\partial\, {}^{(3)}_{h_3}w_{kj}} \frac{1}{H_2} \sum_{h_2=1}^{H_2} \left\{ \left[{}^{(3)(2)}_{h_3\;h_2} \right] \phi(\mathbf{x_i}) \cdot \frac{1}{H_1} \sum_{h_1=1}^{H_1} \left\{ \left[{}^{(2)(1)}_{h_2\;h_1} \right] \phi(\mathbf{x_i}) \cdot \left[{}^{(1)(0)}_{h_1\;\;l} \right] \phi(\mathbf{x_i}) \right\} \right\}. \tag{19}$$

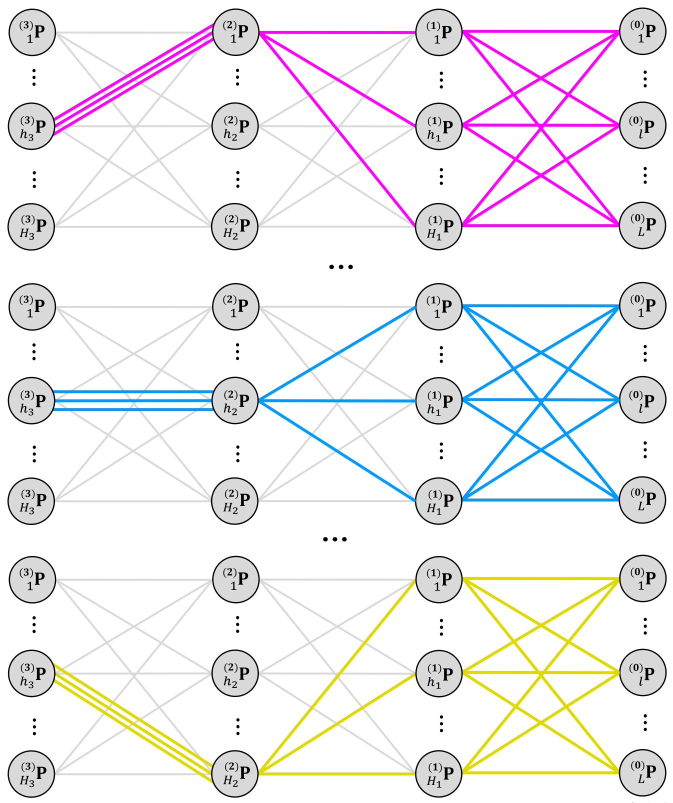


Fig. 5. Representation of all the paths to be considered for the calculation of $\frac{\partial \hat{y}_l(\mathbf{x_i})}{\partial\, {}^{(3)}_{h_3}w_{kj}}$.

Generalizing the solution (17), we can then conclude that

$$\frac{\partial \hat{y}_l(\mathbf{x_i})}{\partial\, {}^{(\lambda)}_{\alpha}w_{kj}} = \frac{\partial\, {}^{(\lambda)}t_{\alpha}(\mathbf{x_i})}{\partial\, {}^{(\lambda)}_{\alpha}w_{kj}} \prod_{\pi=0}^{\lambda-1} \left[{}^{(\pi\,+\,1)(\pi)}_{\;\;\delta\qquad\gamma} \right] \phi(\mathbf{x_i}), \tag{20}$$

$$\frac{\partial \hat{y}_l(\mathbf{x_i})}{\partial\, {}^{(\lambda)}_{\alpha}w_{kj}} = \frac{\partial\, {}^{(\lambda)}t_{\alpha}(\mathbf{x_i})}{\partial\, {}^{(\lambda)}_{\alpha}w_{kj}} \prod_{\pi=1}^{\lambda-1} \left\{ \left[{}^{(\pi\,+\,1)(\pi)}_{\;\;\delta\qquad\gamma} \right] \phi(\mathbf{x_i}) \right\} \cdot \left[{}^{(1)(0)}_{h_1\;\;l} \right] \phi(\mathbf{x_i}). \tag{21}$$

where γ and δ represent the units within the layers π and $\pi + 1$ respectively, and

$$\begin{aligned}\left[\begin{smallmatrix}(\pi+1)&(\pi)\\ \delta&\gamma\end{smallmatrix}\right]\phi(\mathbf{x_i}) = &\left[\sum_{c=1}^{C}{}^{(\pi)}_{\gamma}\mu_c\left({}^{(\pi+1)}\mathbf{t_i}\right)\right]^{-1}\sum_{c=1}^{C}\Big\{{}^{(\pi)}_{\gamma}\mu_c\left({}^{(\pi+1)}\mathbf{t_i}\right)\\ &\cdot\Big\{-2\,{}^{(\pi)}_{\gamma}\mu_c\left({}^{(\pi+1)}\mathbf{t_i}\right)\cdot\left[{}^{(\pi)}_{\gamma}u_{c\delta}\cdot{}^{(\pi+1)}t_\delta(\mathbf{x_i})+{}^{(\pi)}_{\gamma}v_{c\delta}\right]\cdot{}^{(\pi)}_{\gamma}u_{c\delta}\\ &\cdot\left[{}^{(\pi)}_{\gamma}r_c\left({}^{(\pi+1)}\mathbf{t_i}\right)-{}^{(\pi)}t_\gamma(\mathbf{x_i})\right]+{}^{(\pi)}_{\gamma}m_{c\delta}\cdot{}^{(\pi)}_{\gamma}\varpi_c\left({}^{(\pi+1)}\mathbf{t_i}\right)\Big\}\Big\}.\end{aligned}\tag{22}$$

For the case where π is identifying the last hidden layer (the first layer starting from the left), the inputs that receive the units are indeed the inputs of the system, so ${}^{(\pi+1)}\mathbf{t_i}\big|_{\pi=\Pi}=\mathbf{x_i}$, where Π represents the last hidden layer. The formula (20), aligns with our preliminary reasoning and the expected form (5).

Again we consider the averaging of all the possible paths to correct (20),

$$\frac{\partial\hat{y}_l(\mathbf{x_i})}{\partial\,{}^{(\lambda)}_{\alpha}w_{kj}}=\frac{1}{\prod_{\pi=1}^{\lambda-1}H_\pi}\sum_{h_1,\dots,h_{\lambda-1}=1}^{H_1,\dots,H_{\lambda-1}}\left\{\frac{\partial\,{}^{(\lambda)}t_\alpha(\mathbf{x_i})}{\partial\,{}^{(\lambda)}_{\alpha}w_{kj}}\prod_{\pi=1}^{\lambda-1}\left\{\left[\begin{smallmatrix}(\pi+1)&(\pi)\\ \delta&\gamma\end{smallmatrix}\right]\phi(\mathbf{x_i})\right\}\cdot\left[\begin{smallmatrix}(1)&(0)\\ h_1&l\end{smallmatrix}\right]\phi(\mathbf{x_i})\right\},\tag{23}$$

$$\frac{\partial\hat{y}_l(\mathbf{x_i})}{\partial\,{}^{(\lambda)}_{\alpha}w_{kj}}=\frac{\partial\,{}^{(\lambda)}t_\alpha(\mathbf{x_i})}{\partial\,{}^{(\lambda)}_{\alpha}w_{kj}}\frac{1}{\prod_{\pi=1}^{\lambda-1}H_\pi}\sum_{h_1,\dots,h_{\lambda-1}=1}^{H_1,\dots,H_{\lambda-1}}\left\{\prod_{\pi=1}^{\lambda-1}\left\{\left[\begin{smallmatrix}(\pi+1)&(\pi)\\ \delta&\gamma\end{smallmatrix}\right]\phi(\mathbf{x_i})\right\}\cdot\left[\begin{smallmatrix}(1)&(0)\\ h_1&l\end{smallmatrix}\right]\phi(\mathbf{x_i})\right\}.\tag{24}$$

Note that we cannot replace the sum of the product, $\sum_{h_1,\dots,h_{\lambda-1}=1}^{H_1,\dots,H_{\lambda-1}}\left\{\prod_{\pi=1}^{\lambda-1}\left[\begin{smallmatrix}(\pi+1)&(\pi)\\ \delta&\gamma\end{smallmatrix}\right]\phi(\mathbf{x_i})\right\}$, for the product of the sum, $\prod_{\pi=1}^{\lambda-1}\left\{\sum_{h_\pi=1}^{H_\pi}\left[\begin{smallmatrix}(\pi+1)&(\pi)\\ \delta&\gamma\end{smallmatrix}\right]\phi(\mathbf{x_i})\right\}$, since the terms $\left[\begin{smallmatrix}(\pi+1)&(\pi)\\ \delta&\gamma\end{smallmatrix}\right]\phi$ are not independent, i.e., all the intermediate units considered must create a connected path. Instead, we can group the series in the following way

$$\begin{aligned}&\frac{1}{\prod_{\pi=1}^{\lambda-1}H_\pi}\sum_{h_1,\dots,h_{\lambda-1}=1}^{H_1,\dots,H_{\lambda-1}}\left\{\prod_{\pi=1}^{\lambda-1}\left\{\left[\begin{smallmatrix}(\pi+1)&(\pi)\\ \delta&\gamma\end{smallmatrix}\right]\phi(\mathbf{x_i})\right\}\cdot\left[\begin{smallmatrix}(1)&(0)\\ h_1&l\end{smallmatrix}\right]\phi(\mathbf{x_i})\right\}=\frac{1}{H_{\lambda-1}}\sum_{h_{\lambda-1}=1}^{H_{\lambda-1}}\left\{\left[\begin{smallmatrix}(\lambda)&(\lambda-1)\\ \alpha&\beta\end{smallmatrix}\right]\phi(\mathbf{x_i})\right.\\ &\cdot\left\{\dots\cdot\frac{1}{H_2}\left\{\sum_{h_2=1}^{H_2}\left[\begin{smallmatrix}(3)&(2)\\ h_3&h_2\end{smallmatrix}\right]\phi(\mathbf{x_i})\frac{1}{H_1}\left\{\sum_{h_1=1}^{H_1}\left\{\left[\begin{smallmatrix}(2)&(1)\\ h_2&h_1\end{smallmatrix}\right]\phi(\mathbf{x_i})\cdot\left[\begin{smallmatrix}(1)&(0)\\ h_1&l\end{smallmatrix}\right]\phi(\mathbf{x_i})\right\}\right\}\right\}\right\}\left.\vphantom{\sum}\right\}.\end{aligned}\tag{25}$$

Note that in (25), the term $\left[\begin{smallmatrix}(1)&(0)\\ h_1&l\end{smallmatrix}\right]\phi(\mathbf{x_i})$ is not the argument of a hypothetical final series because all the paths have to end in the output unit l. However, when we study the partial derivative of the loss J with respect to the generic parameter ${}^{(\lambda)}_{\alpha}w_{kj}$, and we integrate the definition of $\frac{\partial\hat{y}_l(\mathbf{x_i})}{\partial\,{}^{(\lambda)}_{\alpha}w_{kj}}$ given by (24), we can see how this new formula has one more series whose argument is indeed $\left[\begin{smallmatrix}(1)&(0)\\ h_1&l\end{smallmatrix}\right]\phi(\mathbf{x_i})$ (and the error term $e_{il}=y_{il}-\hat{y}_l(\mathbf{x_i})$),

$$\frac{\partial J(\mathbf{x_i})}{\partial\,{}^{(\lambda)}_{\alpha}w_{kj}}=-\frac{1}{L}\sum_{l=1}^{L}\left\{[y_{il}-\hat{y}_l(\mathbf{x_i})]\frac{\partial\hat{y}_l(\mathbf{x_i})}{\partial\,{}^{(\lambda)}_{\alpha}w_{kj}}\right\}=-\frac{1}{L}\sum_{l=1}^{L}\left\{e_{il}\frac{\partial\hat{y}_l(\mathbf{x_i})}{\partial\,{}^{(\lambda)}_{\alpha}w_{kj}}\right\}=\tag{26}$$

$$= -\frac{\partial^{(\lambda)} t_\alpha(\mathbf{x_i})}{\partial^{(\lambda)}_{\alpha} w_{kj}} \frac{1}{\prod_{\pi=1}^{\lambda-1} H_\pi} \sum_{h_1,\dots,h_{\lambda-1}=1}^{H_1,\dots,H_{\lambda-1}} \left\{ \prod_{\pi=1}^{\lambda-1} \left\{ \begin{bmatrix} (\pi+1)(\pi) \\ \delta \quad \gamma \end{bmatrix} \phi(\mathbf{x_i}) \right\} \cdot \frac{1}{L} \sum_{l=1}^{L} \left\{ \begin{bmatrix} (1)(0) \\ h_1 \quad l \end{bmatrix} \phi(\mathbf{x_i}) \cdot e_{il} \right\} \right\}.$$

So, following with the development of (25), we can replace the term $\begin{bmatrix} (1)(0) \\ h_1 \quad l \end{bmatrix} \phi(\mathbf{x_i})$ for $\sum_{l=1}^{L} \left\{ \begin{bmatrix} (1)(0) \\ h_1 \quad l \end{bmatrix} \phi(\mathbf{x_i}) \cdot e_{il} \right\}$ in order to obtain the definition of $\frac{\partial J(\mathbf{x_i})}{\partial^{(\lambda)}_{\alpha} w_{kj}}$,

$$\frac{1}{\prod_{\pi=1}^{\lambda-1} H_\pi} \sum_{h_1,\dots,h_{\lambda-1}=1}^{H_1,\dots,H_{\lambda-1}} \left\{ \prod_{\pi=1}^{\lambda-1} \left\{ \begin{bmatrix} (\pi+1)(\pi) \\ \delta \quad \gamma \end{bmatrix} \phi(\mathbf{x_i}) \right\} \cdot \sum_{l=1}^{L} \left\{ \begin{bmatrix} (1)(0) \\ h_1 \quad l \end{bmatrix} \phi(\mathbf{x_i}) \cdot e_{il} \right\} \right\} =$$

$$\frac{1}{H_{\lambda-1}} \sum_{h_{\lambda-1}=1}^{H_{\lambda-1}} \left\{ \begin{bmatrix} (\lambda)(\lambda-1) \\ \alpha \quad \beta \end{bmatrix} \phi(\mathbf{x_i}) \right. \tag{27}$$

$$\left. \cdot \left\{ \dots \cdot \frac{1}{H_2} \left\{ \sum_{h_2=1}^{H_2} \begin{bmatrix} (3)(2) \\ h_3 \; h_2 \end{bmatrix} \phi(\mathbf{x_i}) \frac{1}{H_1} \left\{ \sum_{h_1=1}^{H_1} \left\{ \begin{bmatrix} (2)(1) \\ h_2 \; h_1 \end{bmatrix} \phi(\mathbf{x_i}) \cdot \frac{1}{L} \sum_{l=1}^{L} \left\{ \begin{bmatrix} (1)(0) \\ h_1 \quad l \end{bmatrix} \phi(\mathbf{x_i}) \cdot e_{il} \right\} \right\} \right\} \right\} \right\} \right\}.$$

At this point, we can define a recursive function as follows,

$$\begin{cases} \overset{(\pi+1)}{\underset{\delta}{\Gamma}}(\mathbf{x_i}) = \frac{1}{H_\pi} \sum_{\gamma=1}^{H_\pi} \left\{ \begin{bmatrix} (\pi+1)(\pi) \\ \delta \quad \gamma \end{bmatrix} \phi(\mathbf{x_i}) \cdot \overset{(\pi)}{\underset{\gamma}{\Gamma}}(\mathbf{x_i}) \right\} \quad \textbf{if} \quad \pi > -1 \\ \overset{(0)}{\underset{l}{\Gamma}}(\mathbf{x_i}) = e_{il} = y_{il} - \hat{y}_l(\mathbf{x_i}) \end{cases} \tag{28}$$

so that

$$\overset{(1)}{\underset{h_1}{\Gamma}}(\mathbf{x_i}) = \frac{1}{L} \sum_{l=1}^{L} \left\{ \begin{bmatrix} (1)(0) \\ h_1 \quad l \end{bmatrix} \phi(\mathbf{x_i}) \cdot \overset{(0)}{\underset{l}{\Gamma}}(\mathbf{x_i}) \right\} = \frac{1}{L} \sum_{l=1}^{L} \left\{ \begin{bmatrix} (1)(0) \\ h_1 \quad l \end{bmatrix} \phi(\mathbf{x_i}) \cdot e_{il} \right\}. \tag{29}$$

Thus, using the recursive function defined at (28), we can redefine equation (27) as

$$\frac{1}{\prod_{\pi=1}^{\lambda-1} H_\pi} \sum_{h_1,\dots,h_{\lambda-1}=1}^{H_1,\dots,H_{\lambda-1}} \left\{ \prod_{\pi=1}^{\lambda-1} \left\{ \begin{bmatrix} (\pi+1)(\pi) \\ \delta \quad \gamma \end{bmatrix} \phi(\mathbf{x_i}) \right\} \cdot \sum_{l=1}^{L} \left\{ \begin{bmatrix} (1)(0) \\ h_1 \quad l \end{bmatrix} \phi(\mathbf{x_i}) \cdot e_{il} \right\} \right\} = \overset{(\lambda)}{\underset{\alpha}{\Gamma}}(\mathbf{x_i}). \tag{30}$$

Finally, the formula of $\frac{\partial J(\mathbf{x_i})}{\partial^{(\lambda)}_{\alpha} w_{kj}}$ becomes

$$\frac{\partial J(\mathbf{x_i})}{\partial^{(\lambda)}_{\alpha} w_{kj}} = -\overset{(\lambda)}{\underset{\alpha}{\Gamma}}(\mathbf{x_i}) \frac{\partial^{(\lambda)} t_\alpha(\mathbf{x_i})}{\partial^{(\lambda)}_{\alpha} w_{kj}}. \tag{31}$$

This is coherent with what we have for the case of a single hidden layer, i.e., equation (1). Note that $\underset{\alpha}{\overset{(\lambda)}{\Gamma}}(\mathbf{x_i})$ does not depend on the cluster k nor on the dimension j despite the fact that the parameter ${}^{(\lambda)}_{\alpha}w_{kj}$ of equation (31) does.

The calculation of $\frac{\partial\, {}^{(\lambda)}t_\alpha(\mathbf{x_i})}{\partial\, {}^{(\lambda)}_{\alpha}w_{kj}}$ is similar to that studied in [7-8] but instead of having the real input $\mathbf{x_i}$ as input of the layer, we have ${}^{(\lambda+1)}\mathbf{t}(\mathbf{x_i})$, i.e., we substitute x_{ij} for ${}^{(\lambda+1)}t_j(\mathbf{x_i})$. We also denote ${}^{(1)}_{h}\xi_k(\mathbf{x_i}) \overset{\text{def}}{=} {}^{(1)}_{h}\mu_k(\mathbf{x_i}) \cdot \left[\sum_{c=1}^{C} {}^{(1)}_{h}\mu_c(\mathbf{x_i})\right]^{-1}$. After a short straight-forward development, we obtain the partial derivatives of ${}^{(\lambda)}t_\alpha(\mathbf{x_i})$,

$$\frac{\partial\, {}^{(\lambda)}t_\alpha(\mathbf{x_i})}{\partial\, {}^{(\lambda)}_{\alpha}m_{kj}} = {}^{(\lambda)}_{\alpha}\xi_k(\mathbf{x_i}) \cdot {}^{(\lambda)}_{\alpha}\varpi_k(\mathbf{x_i}) \cdot {}^{(\lambda+1)}t_j(\mathbf{x_i}), \tag{32}$$

$$\frac{\partial\, {}^{(\lambda)}t_\alpha(\mathbf{x_i})}{\partial\, {}^{(\lambda)}_{\alpha}n_{k}} = {}^{(\lambda)}_{\alpha}\xi_k(\mathbf{x_i}) \cdot {}^{(\lambda)}_{\alpha}\varpi_k(\mathbf{x_i}), \tag{33}$$

$$\begin{aligned}\frac{\partial\, {}^{(\lambda)}t_\alpha(\mathbf{x_i})}{\partial\, {}^{(\lambda)}_{\alpha}u_{kj}} = -2\, {}^{(\lambda)}_{\alpha}\xi_k(\mathbf{x_i}) \cdot {}^{(\lambda)}_{\alpha}\mu_k(\mathbf{x_i}) \cdot \left[{}^{(\lambda)}_{\alpha}r_k(\mathbf{x_i}) - {}^{(\lambda)}t_\alpha(\mathbf{x_i})\right] \cdot \left({}^{(\lambda)}_{\alpha}u_{kj} \cdot {}^{(\lambda+1)}t_j(\mathbf{x_i}) + {}^{(\lambda)}_{\alpha}v_{kj}\right)\\ \cdot {}^{(\lambda+1)}t_j(\mathbf{x_i}),\end{aligned} \tag{34}$$

$$\frac{\partial\, {}^{(\lambda)}t_\alpha(\mathbf{x_i})}{\partial\, {}^{(\lambda)}_{\alpha}v_{kj}} = -2\, {}^{(\lambda)}_{\alpha}\xi_k(\mathbf{x_i}) \cdot {}^{(\lambda)}_{\alpha}\mu_k(\mathbf{x_i}) \cdot \left[{}^{(\lambda)}_{\alpha}r_k(\mathbf{x_i}) - {}^{(\lambda)}t_\alpha(\mathbf{x_i})\right] \cdot \left({}^{(\lambda)}_{\alpha}u_{kj} \cdot {}^{(\lambda+1)}t_j(\mathbf{x_i}) + {}^{(\lambda)}_{\alpha}v_{kj}\right). \tag{35}$$

Using the learning rate and (31) we can then get

$$\Delta\, {}^{(\lambda)}_{\alpha}w_{kj} = -\eta \frac{\partial J(\mathbf{x_i})}{\partial\, {}^{(\lambda)}_{\alpha}w_{kj}} = \eta \cdot \underset{\alpha}{\overset{(\lambda)}{\Gamma}}(\mathbf{x_i}) \frac{\partial\, {}^{(\lambda)}t_\alpha(\mathbf{x_i})}{\partial\, {}^{(\lambda)}_{\alpha}w_{kj}}, \tag{36}$$

where the update rule of ${}^{(\lambda)}_{\alpha}w_{kj}$ is as follows,

$${}^{(\lambda)}_{\alpha}w_{kj}^{new} = \Delta\, {}^{(\lambda)}_{\alpha}w_{kj} + {}^{(\lambda)}_{\alpha}w_{kj}^{old}. \tag{37}$$

3 On How to Perform the Update While Minimizing the Calculations

The update of the parameters requires the calculation of all the combinations of the paths that link the unit of the parameter and the output units. This would be a heavy calculation if it had to be recalculated at each layer. However, if we are performing the update from right to left (starting with the output layer), then we can recycle the calculations as it happens in the backpropagation method of neural networks.

In this section we explain how this recycling works and which variables should be stored at each layer in order to be retrieved for the updates of the consecutive layers.

First, we need to input an observation $(\mathbf{x_i}, \mathbf{y_i})$ and calculate the prediction with the current parameters of the CEFYDRA, $\hat{\mathbf{y}}_\mathbf{i} = f(\mathbf{x_i})$. Then, we start with the backpropagation phase. As mentioned, we move from the right to left.

We start with the calculation of all the errors of the output units $\{e_{i1}, e_{i2}, \dots, e_{iL}\}$ and we store these values.

We move on to the first hidden layer, and we go through each of the units within the layer. At each unit, we obtain all the $\phi(\mathbf{x_i})$ variables that link the current unit of the hidden layer with all the units of the output layer, calculating the value of $\overset{(1)}{\underset{h_1}{\Gamma}}(\mathbf{x_i}) = \frac{1}{L}\sum_{l=1}^{L}\left\{e_{il} \cdot \begin{bmatrix}(1)(0)\\ h_1 \quad l\end{bmatrix}\phi(\mathbf{x_i})\right\}$. Note that we do not need to recalculate the errors at each $\overset{(1)}{\underset{h_1}{\Gamma}}(\mathbf{x_i})$; instead we retrieve the values from the memory. Then, we store the values of $\left\{\overset{(1)}{\underset{1}{\Gamma}}(\mathbf{x_i}), \overset{(1)}{\underset{2}{\Gamma}}(\mathbf{x_i}), \dots, \overset{(1)}{\underset{H_1}{\Gamma}}(\mathbf{x_i})\right\}$ so that we can retrieve them in the next layer. The time complexity of this step is $O(H_1 \times L)$, and the space complexity $O(H_1)$.

Before moving on to the next layer, we calculate the partial derivatives $\frac{\partial\, {}^{(1)}t_{h_1}(\mathbf{x_i})}{\partial\, {}^{(1)}_{h_1}w_{kj}}$ for all the hidden units of the first hidden layer, for all the parameters in each unit. In a fully connected network, at each unit of the first hidden layer we have a total of $C \times [(H_2 \times 3) + 1]$ parameters, where C is the number of clusters, H_2 the number of input dimensions in the first hidden layer, 3 represents the variables ${}^{(1)}_{h_1}u_{kj}$, ${}^{(1)}_{h_1}v_{kj}$, and ${}^{(1)}_{h_1}m_{kj}$, and 1 represents the independent variable ${}^{(1)}_{h_1}n_k$. Finally, we get the values of $\frac{\partial J(\mathbf{x_i})}{\partial\, {}^{(1)}_{h_1}w_{kj}}$ and then using the learning rates we calculate the deltas $\Delta\, {}^{(1)}_{h_1}w_{kj}$, which need to be stored for the update phase. The time complexity of this step is $O(H_1 \times C \times [(H_2 \times 3) + 1])$ (if all the units have the same number of clusters), and the space complexity is $O(H_1 \times C \times [(H_2 \times 3) + 1])$ which comes from the storage of the deltas.

Next, we move to the second hidden layer. At this point, the process is the same with the exception that for the calculation of $\left\{\overset{(2)}{\underset{1}{\Gamma}}(\mathbf{x_i}), \overset{(2)}{\underset{2}{\Gamma}}(\mathbf{x_i}), \dots, \overset{(2)}{\underset{H_2}{\Gamma}}(\mathbf{x_i})\right\}$, instead of using the error terms as we did for the first hidden layer, we use the values of $\left\{\overset{(1)}{\underset{1}{\Gamma}}(\mathbf{x_i}), \overset{(1)}{\underset{2}{\Gamma}}(\mathbf{x_i}), \dots, \overset{(1)}{\underset{H_1}{\Gamma}}(\mathbf{x_i})\right\}$.

The Algorithms 1, 2, and 3 provide the pseudocode for all the three phases, prediction, backpropagation, and update, respectively.

Algorithm 1: Prediction of $\hat{\mathbf{y}}_\mathbf{i} = f(\mathbf{x}_\mathbf{i})$ where $f(\cdot)$ is the entire CEYDRA. Time complexity $O(\Pi \times H_\lambda)$ where $H_\lambda = \max_{\pi\in\{0,1,\dots,\Pi\}} H_\pi$. Space complexity $O(\Pi \times H_\lambda)$.

```
Input: The observation vector $\mathbf{x}_\mathbf{i}$ of dimension $N$.
Output: The prediction vector $\hat{\mathbf{y}}_\mathbf{i}$ of dimension $L$.

$^{(\Pi+1)}\mathbf{t}_\mathbf{i} \leftarrow \mathbf{x}_\mathbf{i}$
Descending order of layers: $\boldsymbol{\Lambda} \leftarrow \{\Pi, \Pi-1, \dots, 1, 0\}$
for each layer $\lambda$ in $\Lambda$ do
    for each unit $\alpha$ in layer $\lambda$ do
        ${}^{(\lambda)}_{\alpha}\mathbf{P}(\mathbf{t}_\mathbf{i}) \leftarrow \left[\sum_{c=1}^{C} {}^{(\lambda)}_{\alpha}\mu_c\left({}^{(\lambda+1)}\mathbf{t}_\mathbf{i}\right) \cdot {}^{(\lambda)}_{\alpha}r_c\left({}^{(\lambda+1)}\mathbf{t}_\mathbf{i}\right)\right] \cdot \left[\sum_{c=1}^{C} {}^{(\lambda)}_{\alpha}\mu_c\left({}^{(\lambda+1)}\mathbf{t}_\mathbf{i}\right)\right]^{-1}$
        ${}^{(\lambda)}_{\alpha}t_i \leftarrow {}^{(\lambda)}_{\alpha}\mathbf{P}(\mathbf{t}_\mathbf{i})$
        ${}^{(\lambda)}\mathbf{t}_\mathbf{i}[\alpha] \leftarrow {}^{(\lambda)}_{\alpha}t_i$
    end
end
$\hat{\mathbf{y}}_\mathbf{i} \leftarrow {}^{(0)}\mathbf{t}_\mathbf{i}$
return $\hat{\mathbf{y}}_\mathbf{i}$
```

Algorithm 2: Backpropagation process to calculate the values of $\Delta{}^{(\lambda)}_{\alpha}w_{kj}$ of the CEFYDRA. The time complexity is $O\big(\Pi \times H_\lambda \times (H_{\lambda-1} + C \times H_{\lambda+1})\big)$ where $H_\lambda \times H_{\lambda-1} = \max_{\pi\in\{1,\dots,\Pi\}} H_\pi \times H_{\pi-1}$ or $H_{\lambda+1} \times H_\lambda = \max_{\pi\in\{0,1,\dots,\Pi\}} H_{\pi+1} \times H_\pi$, thus we can simplify the previous expressions to $O(\Pi \times C \times H_{\lambda+1} \times H_\lambda)$ with $H_{\lambda+1} \times H_\lambda = \max_{\pi\in\{0,1,\dots,\Pi\}} H_{\pi+1} \times H_\pi$. The space complexity is also $O(\Pi \times C \times H_{\lambda+1} \times H_\lambda)$, which comes from the space needed by $\Delta\mathbf{W}$.

```
Inputs: $\mathbf{x}_\mathbf{i}, \mathbf{y}_\mathbf{i}, \hat{\mathbf{y}}_\mathbf{i}$
Outputs: The update structure $\Delta\mathbf{W}$, organized in a [Layer, Unit] fashion,
which has as many entries as the number of units in the system, and each of
those entries $\Delta\mathbf{W}[\lambda, \alpha]$ is a three-dimensional matrix that stores the values of
$\Delta{}^{(\lambda)}_{\alpha}w_{kj}$.

Initialize with null values: $\Delta\mathbf{W} \leftarrow \mathbf{0}$
for each unit $l$ in layer 0 do
    $\overset{(0)}{\underset{l}{\Gamma}}(\mathbf{x}_\mathbf{i}) \leftarrow y_{il} - \hat{y}_l(\mathbf{x}_\mathbf{i})$
end
Ascending order of layers: $\boldsymbol{\Lambda} \leftarrow \{1, \dots, \Pi-1, \Pi\}$;
for each layer $\lambda$ in $\Lambda$ do
    for each unit $\alpha$ in layer $\lambda$ do
```

```
8           for each unit $\beta$ in layer $\lambda - 1$ do
9               $\begin{bmatrix} {}^{(\lambda)}_{\alpha} & {}^{(\lambda-1)}_{\beta} \end{bmatrix} \phi(\mathbf{x_i}) \leftarrow \left[ \sum_{c=1}^{C} {}^{(\lambda-1)}_{\beta}\mu_c\left({}^{(\lambda)}\mathbf{t_i}\right) \right]^{-1} \sum_{c=1}^{C} \Big\{ {}^{(\lambda-1)}_{\beta}\mu_c\left({}^{(\lambda)}\mathbf{t_i}\right)$
                $\cdot \Big\{ -2\,{}^{(\lambda-1)}_{\beta}\mu_c\left({}^{(\lambda)}\mathbf{t_i}\right) \cdot \left[ {}^{(\lambda-1)}_{\beta}u_{c\alpha} \cdot {}^{(\lambda)}t_\alpha(\mathbf{x_i}) + {}^{(\lambda-1)}_{\beta}v_{c\alpha} \right] \cdot {}^{(\lambda-1)}_{\beta}u_{c\alpha}$
                  $\cdot \left[ {}^{(\lambda-1)}_{\beta}r_c\left({}^{(\lambda)}\mathbf{t_i}\right) - {}^{(\lambda-1)}t_\beta(\mathbf{x_i}) \right] + {}^{(\lambda-1)}_{\beta}m_{c\alpha} \cdot {}^{(\lambda-1)}_{\beta}\varpi_k\left({}^{(\lambda)}\mathbf{t_i}\right) \Big\} \Big\}$
10          end
11          $H_\lambda \leftarrow$ len(layer $\lambda$)
12          $\overset{(\lambda)}{\underset{\alpha}{\Gamma}}(\mathbf{x_i}) \leftarrow \frac{1}{H_{\lambda-1}} \sum_{\beta=1}^{H_{\lambda-1}} \left\{ \begin{bmatrix} {}^{(\lambda)}_{\alpha} & {}^{(\lambda-1)}_{\beta} \end{bmatrix} \phi(\mathbf{x_i}) \cdot \overset{(\lambda-1)}{\underset{\beta}{\Gamma}}(\mathbf{x_i}) \right\}$
13          for each cluster $k$ in unit $\alpha$ do
14              ${}^{(\lambda)}_{\alpha}\xi_k(\mathbf{x_i}) \leftarrow {}^{(\lambda)}_{\alpha}\mu_k(\mathbf{x_i}) \cdot \left[ \sum_{c=1}^{C} {}^{(\lambda)}_{\alpha}\mu_c(\mathbf{x_i}) \right]^{-1}$
15              $\frac{\partial\, {}^{(\lambda)}t_\alpha(\mathbf{x_i})}{\partial\, {}^{(\lambda)}_{\alpha}n_k} \leftarrow {}^{(\lambda)}_{\alpha}\xi_k(\mathbf{x_i}) \cdot {}^{(\lambda)}_{\alpha}\varpi_k(\mathbf{x_i})$
16              Update the entry of the parameter $n$ in $\mathbf{W}$:
17              $\mathbf{W}[\lambda,\alpha]_{\mathrm{k}}\{n\} \leftarrow \frac{\partial\, {}^{(\lambda)}t_\alpha(\mathbf{x_i})}{\partial\, {}^{(\lambda)}_{\alpha}n_k}$
18              Auxiliar variable 1:
19              ${}^{(\lambda)}_{\alpha}\vartheta_k(\mathbf{x_i}) \leftarrow -2\,{}^{(\lambda)}_{\alpha}\xi_k(\mathbf{x_i}) \cdot {}^{(\lambda)}_{\alpha}\mu_k(\mathbf{x_i}) \cdot \left[ {}^{(\lambda)}_{\alpha}r_k(\mathbf{x_i}) - {}^{(\lambda)}t_\alpha(\mathbf{x_i}) \right]$
20              for each dimension $j$ in ${}^{(\lambda+1)}\mathbf{t}(\mathbf{x_i})$ do
21                  Auxiliar variable 2: ${}^{(\lambda+1)}_{\alpha}\varrho_k(\mathbf{x_i}) \leftarrow {}^{(\lambda)}_{\alpha}u_{kj} \cdot {}^{(\lambda+1)}t_j(\mathbf{x_i}) + {}^{(\lambda)}_{\alpha}v_{kj}$
22                  $\frac{\partial\, {}^{(\lambda)}t_\alpha(\mathbf{x_i})}{\partial\, {}^{(\lambda)}_{\alpha}m_{kj}} \leftarrow {}^{(\lambda+1)}t_j(\mathbf{x_i}) \frac{\partial\, {}^{(\lambda)}t_\alpha(\mathbf{x_i})}{\partial\, {}^{(\lambda)}_{\alpha}n_k}$
23                  $\frac{\partial\, {}^{(\lambda)}t_\alpha(\mathbf{x_i})}{\partial\, {}^{(\lambda)}_{\alpha}v_{kj}} \leftarrow {}^{(\lambda+1)}_{\alpha}\varrho_k(\mathbf{x_i}) \cdot {}^{(\lambda)}_{\alpha}\vartheta_k(\mathbf{x_i})$
24                  $\frac{\partial\, {}^{(\lambda)}t_\alpha(\mathbf{x_i})}{\partial\, {}^{(\lambda)}_{\alpha}u_{kj}} \leftarrow {}^{(\lambda+1)}t_j(\mathbf{x_i}) \frac{\partial\, {}^{(\lambda)}t_\alpha(\mathbf{x_i})}{\partial\, {}^{(\lambda)}_{\alpha}v_{kj}}$
25                  Update the entries in $\Delta\mathbf{W}$:
26                  $\Delta\mathbf{W}[\lambda,\alpha]_{\mathrm{kj}}\{m,a,b\} \leftarrow \left\{ \frac{\partial\, {}^{(\lambda)}t_\alpha(\mathbf{x_i})}{\partial\, {}^{(\lambda)}_{\alpha}m_{kj}}, \frac{\partial\, {}^{(\lambda)}t_\alpha(\mathbf{x_i})}{\partial\, {}^{(\lambda)}_{\alpha}a_{kj}}, \frac{\partial\, {}^{(\lambda)}t_\alpha(\mathbf{x_i})}{\partial\, {}^{(\lambda)}_{\alpha}b_{kj}} \right\}$
27              end
28          end
29          $\Delta\mathbf{W}[\lambda,\alpha] \leftarrow \overset{(\lambda)}{\underset{\alpha}{\Gamma}}(\mathbf{x_i}) \cdot \Delta\mathbf{W}[\lambda,\alpha]$
30      end
31  end
32  $\Delta\mathbf{W} \leftarrow \eta \cdot \Delta\mathbf{W}$
33  return $\Delta\mathbf{W}$
```

Algorithm 3: Update process to calculate the parameters of the next epoch. Time complexity $O(\Pi \times H_\lambda)$ where $H_\lambda = \max_{\pi \in \{0,1,\ldots,\Pi\}} H_\pi$. Space complexity $O(1)$ if we modify the pre-existing structure $\mathbf{W}$.

Inputs: The update structure $\Delta\mathbf{W}$, the initial structure of parameters $\mathbf{W^0}$.
Outputs: The updated parameter final structure $\mathbf{W}^f$.

```
1  Initialize with the initial structure: $\mathbf{W}^f \leftarrow \mathbf{W^0}$
2  Order of the layers: $\boldsymbol{\Lambda} \leftarrow \{\Pi, \Pi - 1, \ldots, 1, 0\}$
3  for each layer $\lambda$ in $\boldsymbol{\Lambda}$ do
4     for each unit $\alpha$ in layer $\lambda$ do
5        $\mathbf{W}^{new}[\lambda, \alpha]_{\mathrm{kj}} \leftarrow \Delta\mathbf{W}[\lambda, \alpha]_{\mathrm{kj}} + \mathbf{W}^{old}[\lambda, \alpha]_{\mathrm{kj}}$
6     end
7  end
8  return $\mathbf{W}^{new}$
```

Both the explainability and the self-reorganization concepts of CEFYDRA will be covered in detail in a different paper. However, we make a note on these two so the reader can understand further the implications of this technique.

The ability to explain the predictions generated (i.e., explainability) is a direct consequence of using membership functions to cluster the information and later approximate with different expressions. At each unit of the CEFYDRA, the dominant individual functions with highest membership values can be collected and aggregated to generate a final linear approximation of the inputs and outputs, which can then be displayed to the end user as an explanation of the prediction. Note that this explanations refer to a specific instance.

The ability for self-reorganization of CEFYDRA comes from the fact that during the learning, the membership functions of some clusters might get reduced to the point where they become negligible. At this point, the network can be modified by suppressing such cluster and reassigning it in a different unit.

4 Conclusions

We proposed an alternative core mathematical mechanism for the units of a neural network. The CEFYDRA is an algorithm that leverages a fuzzy Takagi-Sugeno-Kang inference to map the inputs of each unit via Cauchy membership functions and multidimensional logistic functions. In this paper we focused our efforts on demonstrating the learning formulation for the deep hidden layers of the CEFYDRA. We covered the three laws that make this generalization possible, i.e., Displacement, Substitution and Composition. Finally, we also propose a method to minimize the calculations (induced from the fully connected system) and reduce the complexity in the training.

Acknowledgements

The project that generated these results was supported by a grant from the ”la Caixa” Banking Foundation (ID 100010434), whose code is LCF / BQ / AA19 / 11720045.

References

1. Yang, Z., Zhang, A., Sudjianto, A.: GAMI-Net: An explainable neural network based on generalized additive models with structured interactions. Pattern Recognition, 120 (2021).
2. Lee, Y. Extraction of Competitive Factors in a Competitor Analysis Using an Explainable Neural Network. Neural Processing Letters 53, 1979–1994 (2021).
3. Fauvel, K., Lin, T., Masson, V., Fromont, É., Termier, A.: XCM: An Explainable Convolutional Neural Network for Multivariate Time Series Classification. Mathematics 9 (23), (2021).
4. Sasaki, H., Hidaka, Y., Igarashi, H.: Explainable Deep Neural Network for Design of Electric Motors. IEEE Transactions on Magnetics 57(6), 1-4 (2021).
5. Valizadegan, H. et al.: ExoMiner: A Highly Accurate and Explainable Deep Learning Classifier that Validates 301 New Exoplanets. arXiv:2111.10009 (2021).
6. Kim, M. S., Yun, J. P., Park, P.: An Explainable Convolutional Neural Network for Fault Diagnosis in Linear Motion Guide. In IEEE Transactions on Industrial Informatics, 17 (6), 4036-4045 (2021).
7. Gallego, V., Ríos Insua, D.: Current Advances in Neural Networks. Annual Review of Statistics and Its Application 9(1), (2022).
8. Samek, W., Montavon, G., Lapuschkin, S., Anders, C. J., Müller, K. -R.: Explaining Deep Neural Networks and Beyond: A Review of Methods and Applications. Proceedings of the IEEE 109(3), 247-278 (2021).
9. Viaña, J., Ralescu, S., Cohen, K., Ralescu, A., Kreinovich, V: Localized Learning A Possible Alternative to Current Deep Learning Techniques. In: Melin, P., Castillo, O. (eds.) Studies in Computational Intelligence (2021).
10. Viaña, J., Cohen, K.: Fuzzy-Based, Noise-Resilient, Explainable Algorithm for Regression. In: Explainable AI and Other Applications of Fuzzy Techniques. 1st edn. Springer, Cham (2022).
11. Viaña, J., Ralescu, S., Cohen, K., Ralescu, A., Kreinovich, V.: Extension to multidimensional problems of a fuzzy-based explainable and noise-resilient algorithm”. In: Proceedings of the 14th International Workshop on Constraint Programming and Decision Making CoProd’2021, Szeged, Hungary (2021).
12. Viaña, J., Ralescu, S., Cohen, K., Ralescu, A., Kreinovich, V.: Why Cauchy Membership Functions: Reliability. Advances in Artificial Intelligence and Machine Learning, (To Appear).
13. Viaña, J., Ralescu, S., Cohen, K., Ralescu, A., Kreinovich, V.: Why Cauchy Membership Functions: Efficiency. Advances in Artificial Intelligence and Machine Learning, 1(1), 81-88 (2021).