Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

A multi-class classification model with parametrized target outputs for randomized-based feedforward neural networks

Antonio Manuel Durán-Rosal [a], Aggeo Durán-Fernández [b], Francisco Fernández-Navarro [c,*], Mariano Carbonero-Ruz [a]

[a] *Department of Quantitative Methods, Universidad Loyola Andalucía, Spain*
[b] *U-tad Centro Universitario de Tecnología y Arte Digital, Madrid, Spain*
[c] *Department of Computer Languages and Computer Science, University of Málaga, Spain*

ARTICLE INFO

Article history:
Received 13 July 2022
Received in revised form 28 November 2022
Accepted 4 December 2022
Available online 7 December 2022

Keywords:
Randomized-based Feedforward Neural Networks
Random Vector Functional Link Neural Networks
Extreme Learning Machine
Generalized eigenvalue problem
Classification
Kernel methods

ABSTRACT

Randomized-based Feedforward Neural Networks approach regression and classification (binary and multi-class) problems by minimizing the same optimization problem. Specifically, the model parameters are determined through the ridge regression estimator of the patterns projected in the hidden layer space (randomly generated in its neural network version) for models without direct links and the patterns projected in the hidden layer space along with the original input data for models with direct links. The targets are encoded for the multi-class classification problem according to the 1-of-J encoding (J the number of classes), which implies that the model parameters are estimated to project all the patterns belonging to its corresponding class to one and the remaining to zero. This approach has several drawbacks, which motivated us to propose an alternative optimization model for the framework. In the proposed optimization model, model parameters are estimated for each class so that their patterns are projected to a reference point (also optimized during the process), whereas the remaining patterns (not belonging to that class) are projected as far away as possible from the reference point. The final problem is finally presented as a generalized eigenvalue problem. Four models are then presented: the neural network version of the algorithm and its corresponding kernel version for the neural networks models with and without direct links. In addition, the optimization model has also been implemented in randomization-based multi-layer or deep neural networks. The empirical results obtained by the proposed models were compared to those reported by state-of-the-art models in the correct classification rate and a separability index (which measures the degree of separability in projection terms per class of the patterns belonging to the class of the others). The proposed methods show very competitive performance in the separability index and prediction accuracy compared to the neural networks version of the comparison methods (with and without direct links). Remarkably, the model provides significantly superior performance in deep models with direct links compared to its deep model counterpart.

1. Introduction

In statistics, the Root Mean Square Error (RMSE) is a standard measure of goodness-of-fit in regression models [1]. In those models, the goal is to find a function f (belonging to family of functions $\mathcal{F}$), from a training set $\mathcal{D} = \{(\mathbf{x}_n, y_n), n = 1, \ldots, N\}$, being $\mathbf{x}_n = (x_{n1}, \ldots, x_{nK}) \in \mathbb{R}^K$ the nth training pattern, K the number of attributes of the problem and $y_n \in \mathbb{R}$ the desired output associated to the nth pattern, which is the solution of the following optimization problem:

$$\min_{f\in\mathcal{F}} \left(\sum_{n=1}^{N} (f(\mathbf{x}_n) - y_n)^2 \right).$$

Several promising machine learning models estimate their parameters by minimizing the RMSE of their predictions (estimations) and the desired outputs. For example, Randomized-based Feedforward Neural Networks (RFNNs) generally uses randomly selected fixed weights and biases in the hidden layer(s), and the estimation step, which is only employed in the output layer, is based on the RMSE metric. As pointed out in the literature, this random configuration of weights and bias of the nodes in the hidden layer can lead to a poor exploration of the transformed

* Corresponding author.
E-mail address: fafernandez@uma.es (F. Fernández-Navarro).

space if the number of nodes in the hidden layer is not high enough [2,3], thus giving rise to poor generalization performance. In this context, several authors have proposed hybridizing global optimization techniques (such as Genetic Algorithms or Particle Swarm Optimization) with RFNNs giving rise to Evolutionary RFNNs [4–7]. In those approaches, the parameters associated with the hidden layers are determined through the global optimization technique, and the parameters of the output layer are analytically computed [4,7]. The main disadvantage of those approaches is their increase in computational burden, and their advantages are the gains in performance and the obtaining of simpler models (with fewer hidden neurons) [7].

In recent years, RFNNs have become competitive models for both multi-classification and regression problems [8–12]. They have been extensively used not only on traditional supervised machine learning problems but also on time series prediction [13,14], image classification [15,16], and speech recognition [17]. In the field of RFNNs, there exist different approaches which mainly differ in the architecture of their underlying models. In this study, we will focus on two alternative approaches within the umbrella of RFNNs, which vary in existence or not of direct links from the input to the output: the Extreme Learning Machine (ELM) framework (without direct links) [18,19] and the Random Vector Functional Link (RVFL) networks (with direct links) [20–22].

Consequently, the mapping function f in RFNNs is chosen from the following family of functions for networks models with direct links:

$$\mathcal{F} = \{\mathbf{v}'(.)\boldsymbol{\beta}, \boldsymbol{\beta} \in \mathbb{R}^{D+K}\},$$

where $\mathbf{v} : \mathbb{R}^K \to \mathbb{R}^{D+K}$ includes in the first D components the transformation made by the basis functions and in the last K elements the original input data, the chosen transformation of the input space (the first D components of $\mathbf{v}$) will be denoted as $\mathbf{h} : \mathbb{R}^K \to \mathbb{R}^D$ with D the number of basis functions. Hence, in the case of models without direct links, the transformation function includes only the outputs of the basis functions, and accordingly, f is obtained from the following family:

$$\mathcal{F} = \{\mathbf{h}'(.)\boldsymbol{\beta}, \boldsymbol{\beta} \in \mathbb{R}^{D}\}.$$

The advantages of RFNNs concerning backpropagation (BP) networks are twofold [18]: on the one hand, RFNN-based models can provide better generalization results than BP-based networks. On the other hand, RFNN-based models have a much faster learning speed than BP-based networks. Those advantages are partially obtained thanks to the parameter tuning of those models [11,22]. In the neural version of the model, the parameters associated with the basis functions are randomly determined, and therefore, the estimation of the f function is reduced to the estimation of the $\boldsymbol{\beta}$ parameters, which are computed from the following minimization function:

$$\min_{\boldsymbol{\beta}} \left(\sum_{n=1}^{N} \left(\mathbf{v}'(\mathbf{x}_n)\boldsymbol{\beta} - y_n\right)^2 = \|\mathbf{V}\boldsymbol{\beta} - \mathbf{Y}\|^2 \right), \tag{1}$$

where $\mathbf{Y} = (y_1, y_2 \dots, y_N) \in \mathbb{R}^N$ is the vector of desired outputs, $\mathbf{v}(\mathbf{x}_n) = \mathbf{h}(\mathbf{x}_n) \in \mathbb{R}^D$ in models without direct links, $\mathbf{v}(\mathbf{x}_n) = (\mathbf{h}(\mathbf{x}_n), \mathbf{x}_n) \in \mathbb{R}^{D+K}$ in models with direct links, $\mathbf{V} = (\mathbf{H}) \in \mathbb{R}^{N\times D}$ in models without direct links, $\mathbf{V} = (\mathbf{H}\ \mathbf{X}) \in \mathbb{R}^{N\times(D+K)}$ in models with direct links, $\mathbf{H} \in \mathbb{R}^{N\times D}$ the matrix of the hidden layer outputs, $\mathbf{H} = (\mathbf{h}'(\mathbf{x}_1), \mathbf{h}'(\mathbf{x}_2), \dots, \mathbf{h}'(\mathbf{x}_N))$ and $\mathbf{X} \in \mathbb{R}^{N\times K}$ the matrix with the input data, $\mathbf{X} = (\mathbf{x}'_1, \mathbf{x}'_2, \dots, \mathbf{x}'_N)$. The solution to that optimization problem is:

$$\boldsymbol{\beta} = \left(\mathbf{V}'\mathbf{V}\right)^{-1}\mathbf{V}'\mathbf{Y},$$

where an additional term $\frac{\mathbf{I}}{C}$ is traditionally included within the inverse of the matrix for numerical stability reasons (representing the regularization component), being C a user-defined parameter that weights the importance of the model fitting with respect to the regularization [23,24]. The incorporation of the term $\frac{\mathbf{I}}{C}$ to the inversion of the matrix implies the inclusion of the component $\|\boldsymbol{\beta}\|^2$ to the error function of the problem [11] (with the corresponding C parameter to weight the importance of the two components of the equation).

In this context, several works have been proposed using different forms of regularization in RFNNs. Two types of regularization have been the most common and widely used. On the one hand, L^2-norm regularization adds the squared magnitude of the coefficients as a penalty term to the loss function [19,25]. On the other hand, L^1-norm, also known as LASSO regularization, adds the absolute value of the coefficients as a penalty term to the loss function, shrinking the coefficients of the less important characteristic to zero, thus, leading to models with less complex structure [26,27]. Furthermore, it is possible to combine the grouping effect of the L^2 penalty and the sparsity in the solutions of the L^1-norm in the so-called ElasticNet, to control the complexity of the network and prevent overfitting [28,29]. Several authors have recently proposed new algorithmic procedures for estimating the parameters of RFNNs with different typologies of regularization terms. For example, Yan et al. [30] presented an incremental laplacian for the regularization of ELM in semi-supervised online learning problems. In [31], a Lanczos algorithm was proposed to perform a fast regularization to generate a robust outcome without overfitting. In the field of RVFL, a novel fixed iterative algorithm for incorporating the two typologies of regularization terms within the optimization problem of RVFL networks has been recently proposed in [32].

RFNNs-based models address both regression and classification problems through the same optimization problem (Eq. (1)) [19]. The main difference between the two problems lies in the encoding adopted for the targets in classification problems. Thus, the vector with the targets $\mathbf{Y}$ is represented in matrix form in classification problems as $\mathbf{Y} = (\mathbf{Y}_1\ \mathbf{Y}_2\ \dots\ \mathbf{Y}_J) = \begin{pmatrix} \mathbf{y}'_1 \\ \vdots \\ \mathbf{y}'_N \end{pmatrix} \in \mathbb{R}^{N\times J}$, where J is the number of classes, $\mathbf{y}_n \in \{0, 1\}^J$ is the class label assuming the "1-of-J" encoding ($y_{nj} = 1$ if $\mathbf{x}_n$ is a pattern of the jth class, $y_{nj} = 0$ otherwise) and $\mathbf{Y}_j \in \mathbb{R}^N$ is the jth column of the $\mathbf{Y}$ matrix. Additionally, the output matrix of coefficients for classification problems has the form $\boldsymbol{\beta} = (\boldsymbol{\beta}_1\ \boldsymbol{\beta}_2\ \dots\ \boldsymbol{\beta}_J) \in \mathbb{R}^{D\times J}$ being $\boldsymbol{\beta}_j \in \mathbb{R}^D$ the weights of the jth output node in ELM-based models and the form $\boldsymbol{\beta} = (\boldsymbol{\beta}_1\ \boldsymbol{\beta}_2\ \dots\ \boldsymbol{\beta}_J) \in \mathbb{R}^{(D+K)\times J}$ with $\boldsymbol{\beta}_j \in \mathbb{R}^{D+K}$ in RVFL-based models. In this way, RFNN-based models minimizes the following optimization problem in classification problems [19]:

$$\min_{\boldsymbol{\beta}\in\mathbb{R}^{D\times J}} \left(\|\boldsymbol{\beta}\|^2 + C\|\mathbf{V}\boldsymbol{\beta} - \mathbf{Y}\|^2 \right). \tag{2}$$

Although the classification problem is traditionally presented in its matrix form, the final optimization problem is the sum of J separable vector problems, one per class. In fact, if we disaggregate the error function by columns, it can be verified that:

$$\|\boldsymbol{\beta}\|^2 + C\|\mathbf{V}\boldsymbol{\beta} - \mathbf{Y}\|^2 = \sum_{j=1}^{J} \left(\|\boldsymbol{\beta}_j\|^2 + C\|\mathbf{V}\boldsymbol{\beta}_j - \mathbf{Y}_j\|^2 \right).$$

The error function for each class can be reformulated as:

$$\begin{aligned} \|\boldsymbol{\beta}_j\|^2 + C\|\mathbf{V}\boldsymbol{\beta}_j - \mathbf{Y}_j\|^2 &= \boldsymbol{\beta}'_j\boldsymbol{\beta}_j + C(\boldsymbol{\beta}'_j\mathbf{V}' - \mathbf{Y}'_j)(\mathbf{V}\boldsymbol{\beta}_j - \mathbf{Y}_j) \\ &= \boldsymbol{\beta}'_j\boldsymbol{\beta}_j + C\boldsymbol{\beta}'_j\mathbf{V}'\mathbf{V}\boldsymbol{\beta}_j - 2C\mathbf{Y}'_j\mathbf{V}\boldsymbol{\beta}_j + C\mathbf{Y}'_j\mathbf{Y}_j. \end{aligned}$$

Hence, the optimization problem can be rewritten for each class as:

$$\min_{\boldsymbol{\beta}_j} \left(\boldsymbol{\beta}_j'(\boldsymbol{I} + C\mathbf{V}'\mathbf{V})\boldsymbol{\beta}_j - 2C\mathbf{Y}_j'\mathbf{V}\boldsymbol{\beta}_j \right),$$

where $C\mathbf{Y}_j'\mathbf{Y}_j$ is constant. In addition, the component $\|\mathbf{V}\boldsymbol{\beta}_j - \mathbf{Y}_j\|^2$ associated with the jth class could be also rewritten as:

$$\begin{aligned}\|\mathbf{V}\boldsymbol{\beta}_j - \mathbf{Y}_j\|^2 &= \sum_{n=1}^{N} \left(\mathbf{v}'(\mathbf{x}_n)\boldsymbol{\beta}_j - y_{nj}\right)^2 \\ &= \sum_{\mathbf{x}\in\mathcal{C}_j} \left(\mathbf{v}'(\mathbf{x})\boldsymbol{\beta}_j - 1\right)^2 + \sum_{\mathbf{x}\notin\mathcal{C}_j} \left(\mathbf{v}'(\mathbf{x})\boldsymbol{\beta}_j\right)^2,\end{aligned}$$

where $\mathbf{x} \in \mathcal{C}_j$ represents that the training pattern $\mathbf{x}$ belongs to the jth class, $\mathcal{C}_j$. The above expression indicates that the vector $\boldsymbol{\beta}_j$ is estimated aiming to project the patterns of its class to 1 and the others to 0.

As can be seen, the current formulation of the classification problem for RFNN-based models has the following drawbacks:

- It forces training patterns that do not belong to the jth class to be projected to the same point (zero), regardless of their belonging class.
- The selection of the projections points is entirely arbitrary. Furthermore, the distance of the projection between different classes is also arbitrary (and fixed to one).

Motivated by the previously described drawbacks of the error function formulation of the RFNN problem for classification, a novel RFNN classification model is proposed in this manuscript. In the proposed method, the training patterns of the jth class are projected to a projection point that is also estimated during the training stage, β_{0j}, while the remaining ones are projected as far away to this point as possible. It is important to stress that traditional RFNN-based approaches do not consider the bias in the output layer in their implementations, which prompts the separating hyperplane to pass through the origin in random basis functions space [19]. In the proposed method, the bias of the output layer is the corresponding projection point which is also optimized during the optimization stage. In this way, the different hyperplanes per class should all pass through their corresponding projection points instead of the origin.

The problem is formulated as a quadratic fractional programming problem and solved as a generalized eigenvalue problem. The generalized problem has appeared as the solution of an umbrella of SVM-based models that aims to estimate non-parallel optimal hyperplanes to overcome the well-documented limitation of SVM models based on parallel hyperplanes when addressing cross data [33–40]. In this manuscript, those ideas have been explored from the RFNN perspective, aiming to improve the performance of RFNN-based models, including accuracy or training time, with the final goal of proposing a novel model that addresses the multiclass classification problem without implementing the 1-versus-one and 1-versus-all approaches and where the targets per class are estimated during the optimization procedure. The manuscript presents three versions of the model for each architecture (with and without direct links): the standard formulation, a formulation including regularization, and a kernelized version of the model.

The manuscript is organized as follows: an explanation of our proposal is discussed in Section 2. The experimental framework is presented in Section 3, and the empirical comparisons are in Section 4. Conclusions and discussion are in the final segment of the article, Section 5.

2. The proposed method

This section aims to mathematically describe the architecture of the proposed model, the formulation of its error function formulation, the matrix formulation of the problem, and how its parameters are obtained. As previously detailed, the models could be implemented in RFNNs without and with direct links. For the sake of simplicity, we have decided to explain the procedure in networks without direct links fully, and after that, we have detailed the mathematical modifications needed to implement the idea in networks with direct links. Additionally, the algorithmic steps required to estimate the parameters of the six models (three per architecture) included are also fully described, along with the definition of the classification rule of the models.

2.1. Networks without direct links

This section aims to provide all the required details to implement the proposal in RFNNs without direct links. The most important example of RFNNs without direct links is most likely ELM, and for that reason, in some parts of the text, we refer directly to it.

2.1.1. Error function formulation

The goal of the error function proposed is two-fold: firstly, to project as close as possible to the patterns belonging to the jth class to their reference projection point, β_{0j}, and secondly, to project as far away as possible to the reference point β_{0j} associated with the jth class to the patterns that do not belong to the jth class. Thus, the model proposed has to estimate two types of parameters per class: the projection vector $\boldsymbol{\beta}_j$, and the reference projection point, $\beta_{0j}, j = 1, \ldots, J$.

The optimization problem of dimension J is solved by independently estimating the projection vectors $\boldsymbol{\beta}_j$, and the reference projection points, β_{0j}, per class $j = 1, \ldots, J$. The first objective could mathematically be formulated as the minimization of the following expression:

$$\sum_{\mathbf{x}\in\mathcal{C}_j} \left(\mathbf{h}'(\mathbf{x})\boldsymbol{\beta}_j - \beta_{0j}\right)^2. \tag{3}$$

The second objective is that the rest of the patterns are projected as far as possible from the reference point. That is, to maximize the following expression:

$$\sum_{\mathbf{x}\notin\mathcal{C}_j} \left(\mathbf{h}'(\mathbf{x})\boldsymbol{\beta}_j - \beta_{0j}\right)^2. \tag{4}$$

The two objectives can be satisfied jointly if the following expression is minimized:

$$\min_{\boldsymbol{\beta}_j\in\mathbb{R}^D, \beta_{0j}\in\mathbb{R}} \left(\frac{\sum_{\mathbf{x}\in\mathcal{C}_j} \left(\mathbf{h}'(\mathbf{x})\boldsymbol{\beta}_j - \beta_{0j}\right)^2}{\sum_{\mathbf{x}\notin\mathcal{C}_j} \left(\mathbf{h}'(\mathbf{x})\boldsymbol{\beta}_j - \beta_{0j}\right)^2} \right). \tag{5}$$

2.1.2. Matrix formulation

This section aims to rewrite the objective function of the model proposed (defined in Eq. (5)) in matrix form. Let us denote the vector of parameters of the jth class $\mathbf{B}_j$ as $\mathbf{B}_j = \begin{pmatrix} \beta_{0j} \\ \boldsymbol{\beta}_j \end{pmatrix} \in \mathbb{R}^{D+1}$, the matrix $\mathbf{M}_j$ as $\mathbf{M}_j = \sum_{\mathbf{x}\in\mathcal{C}_j} \begin{pmatrix} -1 \\ \mathbf{h}(\mathbf{x}) \end{pmatrix} \left(-1 \; \mathbf{h}'(\mathbf{x})\right) \in \mathbb{R}^{(D+1)\times(D+1)}$ and $\mathbf{M} = \sum_{j=1}^{J} \mathbf{M}_j$. The matrix $\mathbf{M}_j$ includes information about the variance of the outputs of the basis functions, the interaction between basis functions, and the number of training

```
PTELM(D):
Require: Training set: $\mathcal{D} = \{(\mathbf{x}_n, \mathbf{y}_n)\}_{n=1}^N$, where $\mathbf{x}_n \in \mathbb{R}^K$ and $\mathbf{y}_n \in \{0,1\}^J$.
Ensure: Optimized neural network model: $\{\mathbf{B}_1, \ldots, \mathbf{B}_J\}$.
1: for $j = 1$ until $J$ do
2:     $\mathbf{H}_j \leftarrow (\mathbf{h}'(\mathbf{x}), \mathbf{x}_n \in \mathcal{C}_j)$.
3:     $\mathbf{M}_j \leftarrow \begin{pmatrix} N_j & -\sum_{\mathbf{x}\in\mathcal{C}_j}\mathbf{h}'(\mathbf{x}) \\ -\sum_{\mathbf{x}\in\mathcal{C}_j}\mathbf{h}(\mathbf{x}) & \mathbf{H}_j'\mathbf{H}_j \end{pmatrix}$.
4: end for
5: $\mathbf{M} \leftarrow \sum_{j=1}^J \mathbf{M}_j$
6: for $j = 1$ until $J$ do
7:     $\mathbf{B}_j \leftarrow \mathrm{eig}(\mathbf{M}, \mathbf{M}_j)$.
8: end for
9: return $\{\mathbf{B}_1, \ldots, \mathbf{B}_J\}$.
```

Fig. 1. PTELM training algorithm framework.

patterns per class. In fact, the matrix $\mathbf{M}_j$ can be alternatively be expressed as:

$$\mathbf{M}_j = \begin{pmatrix} N_j & -\sum_{\mathbf{x}\in\mathcal{C}_j}\mathbf{h}'(\mathbf{x}) \\ -\sum_{\mathbf{x}\in\mathcal{C}_j}\mathbf{h}(\mathbf{x}) & \mathbf{H}_j'\mathbf{H}_j \end{pmatrix},$$

where N_j is the number of patterns of the jth class and

$$\mathbf{H}_j = \left(\mathbf{h}'(\mathbf{x}), \mathbf{x} \in \mathcal{C}_j\right) \in \mathbb{R}^{N_j \times D},$$

the matrix with the outputs of the basis function for the jth class (including only the training patterns belonging to the jth class). Consequently, the matrix $\mathbf{H}$ can be written as $\mathbf{H} = \begin{pmatrix} \mathbf{H}_1 \\ \ldots \\ \mathbf{H}_J \end{pmatrix} \in \mathbb{R}^{N\times D}$, with $N = \sum_{j=1}^J N_j$.

So, the first objective can be rewritten in matrix form as:

$$\begin{aligned}\sum_{\mathbf{x}\in\mathcal{C}_j}\left(\mathbf{h}'(\mathbf{x})\boldsymbol{\beta}_j - \beta_{0j}\right)^2 &= \sum_{\mathbf{x}\in\mathcal{C}_j}(\left(-1\ \mathbf{h}'(\mathbf{x})\right)\mathbf{B}_j)^2 \\ &= \sum_{\mathbf{x}\in\mathcal{C}_j}\mathbf{B}_j'\begin{pmatrix} -1 \\ \mathbf{h}(\mathbf{x})\end{pmatrix}\left(-1\ \mathbf{h}'(\mathbf{x})\right)\mathbf{B}_j \\ &= \mathbf{B}_j'\mathbf{M}_j\mathbf{B}_j.\end{aligned}$$

The second objective can be rewritten as follows:

$$\begin{aligned}\sum_{\mathbf{x}\notin\mathcal{C}_j}\left(\mathbf{h}'(\mathbf{x})\boldsymbol{\beta}_j - \beta_{0j}\right)^2 &= \sum_{\mathbf{x}\notin\mathcal{C}_j}(\left(-1\ \mathbf{h}'(\mathbf{x})\right)\mathbf{B}_j)^2 \\ &= \sum_{\mathbf{x}\notin\mathcal{C}_j}\mathbf{B}_j'\begin{pmatrix} -1 \\ \mathbf{h}(\mathbf{x})\end{pmatrix}\left(-1\ \mathbf{h}'(\mathbf{x})\right)\mathbf{B}_j \\ &= \mathbf{B}_j'(\mathbf{M}-\mathbf{M}_j)\mathbf{B}_j.\end{aligned}$$

Hence, the optimization problem is defined as:

$$\min_{\mathbf{B}_j\in\mathbb{R}^{D+1}}\left(\frac{\mathbf{B}_j'\mathbf{M}_j\mathbf{B}_j}{\mathbf{B}_j'(\mathbf{M}-\mathbf{M}_j)\mathbf{B}_j}\right), \tag{6}$$

which is equivalent to:

$$\max_{\mathbf{B}_j\in\mathbb{R}^{D+1}}\left(\frac{\mathbf{B}_j'(\mathbf{M}-\mathbf{M}_j)\mathbf{B}_j}{\mathbf{B}_j'\mathbf{M}_j\mathbf{B}_j} = \frac{\mathbf{B}_j'\mathbf{M}\mathbf{B}_j}{\mathbf{B}_j'\mathbf{M}_j\mathbf{B}_j} - 1\right). \tag{7}$$

2.1.3. Analytical solution

The final optimization problem can be rewritten, according to the Rayleigh–Ritz quotient method [41], as:

$$\begin{aligned}&\max_{\mathbf{B}_j\in\mathbb{R}^{D+1}} && \left(\mathbf{B}_j'\mathbf{M}\mathbf{B}_j\right) \\ &\text{s.t.} && \mathbf{B}_j'\mathbf{M}_j\mathbf{B}_j = \mathrm{T},\end{aligned} \tag{8}$$

and can be solved using Lagrange multipliers. The Lagrangian [42] is:

$$\mathcal{L} = \mathbf{B}_j'\mathbf{M}\mathbf{B}_j - \lambda_1(\mathbf{B}_j'\mathbf{M}_j\mathbf{B}_j - \mathrm{T}), \tag{9}$$

where λ_1 is the Lagrange multiplier associated with the first constraint. Equating the derivative of $\mathcal{L}$ to zero gives:

$$\frac{\partial\mathcal{L}}{\partial\mathbf{B}_j} = 2\mathbf{M}\mathbf{B}_j - 2\lambda_1\mathbf{M}_j\mathbf{B}_j = 0, \tag{10}$$

so:

$$\mathbf{M}\mathbf{B}_j = \lambda_1\mathbf{M}_j\mathbf{B}_j, \tag{11}$$

which is a generalized eigenvalue problem $(\mathbf{M}, \mathbf{M}_j)$ that can solved using any eigenvalue routine.

A straightforward way to resolve the problem is by left-multiplying the expressions by $\mathbf{M}_j^{-1}$ (if $\mathbf{M}_j$ is not singular):

$$(\mathbf{M}_j^{-1}\mathbf{M})\mathbf{B}_j = \lambda_1\mathbf{B}_j, \tag{12}$$

which is a simple eigenvalue problem for the matrix $(\mathbf{M}_j^{-1}\mathbf{M})$. Finally, it is important to stress that as Eq. (7) is a maximization problem, the eigenvector associated with the optimal $\mathbf{B}_j$ vector is the one having the largest eigenvalue.

The algorithmic flow of the first version of the model, named Parametrized Targets ELM (PTELM), is shown in Fig. 1, being eig(·) the function that solves the generalized eigenvalue problem.

2.1.4. Regularization

The first version of the algorithm has two implicit drawbacks: (i) the norm of $\mathbf{B}_j$, $\|\mathbf{B}_j\|^2$, depends on the value of the first eigenvalue of the matrix $\mathbf{M}_j$, and therefore, there is no a proper control on it, and (ii) the solution of the system depends on that the matrix $\mathbf{B}_j$ is not singular. Both problems can be addressed by including an additional constraint on the norm of the parameters vector in the optimization problem as follows:

$$\begin{aligned}&\max_{\mathbf{B}_j\in\mathbb{R}^{D+1}} && \left(\mathbf{B}_j'\mathbf{M}\mathbf{B}_j\right) \\ &\text{s.t.} && \mathbf{B}_j'\mathbf{M}_j\mathbf{B}_j = \mathrm{T} \\ &&& \mathbf{B}_j'\mathbf{B}_j = 1.\end{aligned} \tag{13}$$

The Lagrangian of the optimization problem is now defined as:

$$\mathcal{L} = \mathbf{B}_j'\mathbf{M}\mathbf{B}_j - \lambda_1(\mathbf{B}_j'\mathbf{M}_j\mathbf{B}_j - \mathrm{T}) - \lambda_2(\mathbf{B}_j'\mathbf{B}_j - 1), \tag{14}$$

and equating the derivative of $\mathcal{L}$ to zero:

$$\frac{\partial\mathcal{L}}{\partial\mathbf{B}_j} = 2\mathbf{M}\mathbf{B}_j - 2\lambda_1\mathbf{M}_j\mathbf{B}_j - 2\lambda_2\mathbf{B}_j = 0. \tag{15}$$

```
PTRELM($D$, $\delta$):
Require: Training set: $\mathcal{D} = \{(\mathbf{x}_n, \mathbf{y}_n)\}_{n=1}^N$, where $\mathbf{x}_n \in \mathbb{R}^K$ and $\mathbf{y}_n \in \{0,1\}^J$.
Ensure: Optimized neural network model: $\{\mathbf{B}_1, \dots, \mathbf{B}_J\}$.
1: for $j = 1$ until $J$ do
2:     $\mathbf{H}_j \leftarrow (\mathbf{h}'(\mathbf{x}), \mathbf{x}_n \in \mathcal{C}_j)$.
3:     $\mathbf{M}_j \leftarrow \begin{pmatrix} N_j & -\sum_{\mathbf{x}\in\mathcal{C}_j}\mathbf{h}'(\mathbf{x}) \\ -\sum_{\mathbf{x}\in\mathcal{C}_j}\mathbf{h}(\mathbf{x}) & \mathbf{H}_j'\mathbf{H}_j \end{pmatrix}$.
4: end for
5: $\mathbf{M} \leftarrow \sum_{j=1}^J \mathbf{M}_j$
6: for $j = 1$ until $J$ do
7:     $\mathbf{B}_j \leftarrow \mathrm{eig}(\mathbf{M}, \mathbf{M}_j + \delta\mathbf{I})$.
8: end for
9: return $\{\mathbf{B}_1, \dots, \mathbf{B}_J\}$.
```

Fig. 2. PTRELM training algorithm framework.

Therefore the solution satisfies that:

$$2(\mathbf{M} - \lambda_1\mathbf{M}_j - \lambda_2\mathbf{I})\mathbf{B}_j = 0, \tag{16}$$

or

$$(\mathbf{M} - \lambda_1(\mathbf{M}_j + \delta\mathbf{I}))\mathbf{B}_j = 0, \tag{17}$$

with $\delta = \frac{\lambda_2}{\lambda_1}$. Again, this is a generalized eigenvalue problem with matrices $(\mathbf{M}, \mathbf{M}_j + \delta\mathbf{I})$, and the solution of the system is the eigenvector of largest eigenvalue of $(\mathbf{M}_j + \delta\mathbf{I})^{-1}\mathbf{M}$. As can be seen, the regularization strengthens the main diagonal of the matrix $\mathbf{M}_j$ in order to make it full rank.

The algorithmic flow of model with regularization (Fig. 2), named Parametrized Targets with Regularization ELM (PTRELM), presents the following modifications with respect to its base model, the PTELM model (Fig. 1): (i) it includes two hyper-parameters (the number of basis functions, D, and the importance of the regularization in the model, δ) and (ii) the eigenvalue function routine is called with the matrices $(\mathbf{M}, \mathbf{M}_j + \delta\mathbf{I})$.

2.1.5. *Kernel version*

In the kernel implementation of the model, the $\mathbf{h}(\mathbf{x})$ function is an unknown feature mapping. Explicitly computing the mappings $\mathbf{h}(\mathbf{x})$ can be computationally expensive and, in many cases, intractable. For example, the unknown feature mapping may be infinitely dimensional. Fortunately, there are certain functions $k(\mathbf{x}_i, \mathbf{x}_j)$ that compute the dot product in another space, $k(\mathbf{x}_i, \mathbf{x}_j) = \langle \mathbf{h}(\mathbf{x}_i), \mathbf{h}(\boldsymbol{x}_j)\rangle$ (for all $\mathbf{x}_i$ and $\mathbf{x}_j$ in the input space). For the sake of simplicity, the kernel function implemented in this study is the Gaussian one:

$$k(\boldsymbol{u}, \boldsymbol{v}) = \exp(-\sigma\|\boldsymbol{u} - \boldsymbol{v}\|^2), \tag{18}$$

where $\sigma \in \mathbb{R}$ is the kernel parameter. Thus, we will reformulate the solution to group the $\mathbf{h}(\mathbf{x})$ terms in dot products and apply the kernel trick to them (substituting those elements by their kernel functions), rather than explicitly mapping the training patterns to the feature mapping.

PTRELM can be reformulated in terms of dot products by first noting that $\boldsymbol{\beta}_j$, $j = 1, \dots, J$, will have an expansion of the form [43]:

$$\boldsymbol{\beta}_j = \sum_{n=1}^N \alpha_{jn}\mathbf{h}(\mathbf{x}_n), \tag{19}$$

where $\boldsymbol{\alpha}_j = (\alpha_{j1}, \dots, \alpha_{jN}) \in \mathbb{R}^N$ is the vector of parameters to be estimated associated to the jth class. Thus, the first objective of the optimization problem could be rewritten as:

$$\sum_{\mathbf{x}\in\mathcal{C}_j}\left(\mathbf{h}'(\mathbf{x})\boldsymbol{\beta}_j - \beta_{0j}\right)^2 = \sum_{\mathbf{x}\in\mathcal{C}_j}\left(\mathbf{h}'(\mathbf{x})\sum_{n=1}^N \alpha_{jn}\mathbf{h}(\mathbf{x}_n) - \beta_{0j}\right)^2 = \sum_{\mathbf{x}\in\mathcal{C}_j}\left(\sum_{n=1}^N k(\mathbf{x}, \mathbf{x}_n)\alpha_{jn} - \beta_{0j}\right)^2.$$

Let us denote the vector of parameters of the jth class $\mathbf{A}_j$ as $\mathbf{A}_j = \begin{pmatrix} \beta_{0j} \\ \boldsymbol{\alpha}_j \end{pmatrix} \in \mathbb{R}^{N+1}$ and $\mathbf{K}(\mathbf{x})$ as $\mathbf{K}(\mathbf{x}) = (k(\mathbf{x}, \mathbf{x}_1), \dots, k(\mathbf{x}, \mathbf{x}_N)) \in \mathbb{R}^N$. Additionally, the matrix $\mathbf{P}_j$ is, in this case, defined as:

$$\mathbf{P}_j = \sum_{\mathbf{x}\in\mathcal{C}_j}\begin{pmatrix} -1 \\ \mathbf{K}(\mathbf{x}) \end{pmatrix}\left(-1 \ \mathbf{K}'(\mathbf{x})\right) \in \mathbb{R}^{(N+1)\times(N+1)},$$

with $\mathbf{P} = \sum_{j=1}^J \mathbf{P}_j$. In fact, the matrix $\mathbf{P}_j$ can be alternatively be expressed as:

$$\mathbf{P}_j = \begin{pmatrix} N_j & -\sum_{\mathbf{x}\in\mathcal{C}_j}\mathbf{K}'(\mathbf{x}) \\ -\sum_{\mathbf{x}\in\mathcal{C}_j}\mathbf{K}(\mathbf{x}) & \mathbf{K}_j'\mathbf{K}_j \end{pmatrix} \in \mathbb{R}^{(N+1)\times(N+1)},$$

where

$$\mathbf{K}_j = \left(\mathbf{K}'(\mathbf{x}), \mathbf{x}_n \in \mathcal{C}_j\right) \in \mathbb{R}^{N_j\times N}.$$

Taking this into account, the first objective could be rewritten in matrix form as follows:

$$\sum_{\mathbf{x}\in\mathcal{C}_j}\left(\sum_{n=1}^N k(\mathbf{x}, \mathbf{x}_n)\alpha_{jn} - \beta_{0j}\right)^2 = \sum_{\mathbf{x}\in\mathcal{C}_j}\left(\left(-1 \ \mathbf{K}'(\mathbf{x})\right)\mathbf{A}_j\right)^2 = \sum_{\mathbf{x}\in\mathcal{C}_j}\mathbf{A}_j'\begin{pmatrix} -1 \\ \mathbf{K}(\mathbf{x}) \end{pmatrix}\left(-1 \ \mathbf{K}'(\mathbf{x})\right)\mathbf{A}_j = \mathbf{A}_j'\mathbf{P}_j\mathbf{A}_j.$$

Analogously, the second objective could be reformulated in its kernel version as:

$$\mathbf{A}_j'(\mathbf{P} - \mathbf{P}_j)\mathbf{A}_j, \tag{20}$$

```
KPTRELM($\sigma$, $\delta$):
Require: Training set: $\mathcal{D} = \{(\mathbf{x}_n, \mathbf{y}_n)\}_{n=1}^{N}$, where $\mathbf{x}_n \in \mathbb{R}^K$ and $\mathbf{y}_n \in \{0,1\}^J$.
Ensure: Optimized neural network model: $\{\mathbf{A}_1, \ldots, \mathbf{A}_J\}$.
1: for $j = 1$ until $J$ do
2:     $\mathbf{K}_j \leftarrow (\mathbf{K}'(\mathbf{x}), \mathbf{x}_n \in \mathcal{C}_j)$.
3:     $\mathbf{P}_j \leftarrow \begin{pmatrix} N_j & -\sum_{\mathbf{x}\in\mathcal{C}_j} \mathbf{K}'(\mathbf{x}) \\ -\sum_{\mathbf{x}\in\mathcal{C}_j} \mathbf{K}(\mathbf{x}) & \mathbf{K}_j'\mathbf{K}_j \end{pmatrix}$.
4: end for
5: $\mathbf{P} \leftarrow \sum_{j=1}^{J} \mathbf{P}_j$
6: for $j = 1$ until $J$ do
7:     $\mathbf{A}_j \leftarrow \mathrm{eig}(\mathbf{P}, \mathbf{P}_j + \delta\mathbf{I})$.
8: end for
9: return $\{\mathbf{A}_1, \ldots, \mathbf{A}_J\}$.
```

Fig. 3. KPTRELM training algorithm framework.

and the optimization function in the kernel version of the model as:

$$\max_{\mathbf{A}_j \in \mathbb{R}^{N+1}} \left(\frac{\mathbf{A}_j'(\mathbf{P}-\mathbf{P}_j)\mathbf{A}_j}{\mathbf{A}_j'\mathbf{P}_j\mathbf{A}_j} = \frac{\mathbf{A}_j'\mathbf{P}\mathbf{A}_j}{\mathbf{A}_j'\mathbf{P}_j\mathbf{A}_j} - 1 \right). \tag{21}$$

Again, the solution of the optimization problem (including also regularization) is the eigenvector of the largest eigenvalue of the generalized eigenvalue problem with matrices $(\mathbf{P}, \mathbf{P}_j + \delta\mathbf{I})$.

Hence, the kernelized version of the model with regularization, named Kernel PTRELM (KPTRELM), includes two hyperparameters: the kernel and regularization parameters, σ and δ, and solves the generalized eigenvalue problem associated with matrices $(\mathbf{P}, \mathbf{P}_j + \delta\mathbf{I})$. Finally, Fig. 3 shows the algorithmic flow of the KPTRELM model.

Given the solution for $\boldsymbol{\alpha}_j$ for the jth class, the projection of a new data (test) point, $\mathbf{x}$, is given by:

$$\begin{aligned}\mathbf{h}'(\mathbf{x})\boldsymbol{\beta}_j &= \mathbf{h}'(\mathbf{x})\sum_{n=1}^{N} \mathbf{h}(\mathbf{x}_n)\alpha_{jn} \\ &= \sum_{n=1}^{N} k(\mathbf{x}, \mathbf{x}_n)\alpha_{jn} \\ &= \mathbf{K}'(\mathbf{x})\boldsymbol{\alpha}_j\end{aligned}$$

2.2. Networks with direct links

For the adaptation of the proposal to RFNNs with direct links, it is essential to consider that the input of the output layer of the network includes both the basis functions' outputs and the original input data (as there are direct links from the input to the output layer). For this reason, the parameters to be estimated per class has the dimensions, $\boldsymbol{\beta}_j \in \mathbb{R}^{D+K}$ and $\beta_{0j} \in \mathbb{R}$ and consequently, the objective function is defined as:

$$\min_{\boldsymbol{\beta}_j \in \mathbb{R}^{D+K}, \beta_{0j} \in \mathbb{R}} \left(\frac{\sum_{\mathbf{x}\in\mathcal{C}_j} \left((\mathbf{h}'(\mathbf{x})\ \mathbf{x}')\boldsymbol{\beta}_j - \beta_{0j}\right)^2}{\sum_{\mathbf{x}\notin\mathcal{C}_j} \left((\mathbf{h}'(\mathbf{x})\ \mathbf{x}')\boldsymbol{\beta}_j - \beta_{0j}\right)^2} \right). \tag{22}$$

Accordingly, the vector with the parameters to be estimated is defined as $\mathbf{B}_j = \begin{pmatrix} \beta_{0j} \\ \boldsymbol{\beta}_j \end{pmatrix} \in \mathbb{R}^{D+K+1}$, the design matrix $\mathbf{M}_j$ has the dimension $(D+K+1)\times(D+K+1)$ and it is defined in this case as:

$$\mathbf{M}_j = \begin{pmatrix} N_j & -\sum_{\mathbf{x}\in\mathcal{C}_j} \mathbf{h}'(\mathbf{x}) & -\sum_{\mathbf{x}\in\mathcal{C}_j} \mathbf{x}' \\ -\sum_{\mathbf{x}\in\mathcal{C}_j} \mathbf{h}(\mathbf{x}) & \mathbf{H}_j'\mathbf{H}_j & \mathbf{H}_j'\mathbf{X}_j \\ -\sum_{\mathbf{x}\in\mathcal{C}_j} \mathbf{x} & \mathbf{X}_j'\mathbf{H}_j & \mathbf{X}_j'\mathbf{X}_j \end{pmatrix},$$

with $\mathbf{X}_j = \left(\mathbf{x}', \mathbf{x} \in \mathcal{C}_j\right) \in \mathbb{R}^{N_j \times K}$. As can be seen, the main difference in the design matrix is that the one proposed for models with direct links includes the variances and interactions among the basis functions and the variances and interactions among the original input variables, along with the interactions between input variables and basis functions. In this sense, the design matrix associated with the model for RVFL architectures includes more information about the data of the machine learning problem.

In algorithmic terms, it is vital to clarify that the PTRVFL and the PTRRVFL are the same as their PTELM and PTRELM counterparts with the unique modification of the design matrix $\mathbf{M}_j$, which in the RVFL versions of the model also includes interactions among input patterns and input data with basis functions' outputs.

In the kernel version of the algorithm, named KPTRRVFL, the model includes two types of kernels in its formulation: the gaussian kernel denoted as $k(\boldsymbol{u}, \boldsymbol{v})$, and the linear kernel, denoted as $l(\boldsymbol{u}, \boldsymbol{v})$. The linear kernel is mathematically defined as:

$$l(\boldsymbol{u}, \boldsymbol{v}) = \boldsymbol{u}'\boldsymbol{v}, \tag{23}$$

which is the simplest kernel function and generally yields models equivalent to their non-kernelized counterparts. Additionally, it is also important to define a function associated with the linear kernel which is represented as $\mathbf{L}(\mathbf{x}) = (l(\mathbf{x}, \mathbf{x}_1), \ldots, l(\mathbf{x}, \mathbf{x}_N)) \in \mathbb{R}^N$.

Similarly to what occurs in the RVFL neural versions of the model, the main difference in the kernel version for RVFL lies in the design matrix, $\mathbf{P}_j$, which in this case is defined as:

$$\mathbf{P}_j = \begin{pmatrix} N_j & -\sum_{\mathbf{x}\in\mathcal{C}_j} \mathbf{K}'(\mathbf{x}) + \mathbf{L}'(\mathbf{x}) \\ -\sum_{\mathbf{x}\in\mathcal{C}_j} \mathbf{K}(\mathbf{x}) + \mathbf{L}(\mathbf{x}) & \mathbf{K}_j'\mathbf{K}_j + \mathbf{L}_j'\mathbf{L}_j \end{pmatrix} \in \mathbb{R}^{(N+1)\times(N+1)},$$

with $\mathbf{L}_j = \left(\mathbf{L}'(\mathbf{x}), \mathbf{x}_n \in \mathcal{C}_j\right) \in \mathbb{R}^{N_j \times N}$. As seen in the design matrix of the KPTRRVFL model, the outputs of the gaussian kernel are summed to those of the linear kernel, keeping the dimension of the problem constant. Again, in algorithmic terms, the only difference to its ELM version lies in the design matrix, $\mathbf{P}_j$.

2.3. Classification rule

Once the projection vectors per class and the corresponding reference points have been estimated, it is necessary to determine a criterion for classifying a given test pattern. In this manuscript, the classification criterion will be to assign the pattern to the class that minimizes the following expression:

$$\frac{|\mathbf{v}'(\mathbf{x})\boldsymbol{\beta}_j - \beta_{0j}|}{\|\boldsymbol{\beta}_j\|^2}, \tag{24}$$

for $j = 1, \dots, J$. In this way, the predicted class label for a test pattern $\mathbf{x}$ is stored in a vector $\widehat{y}(\mathbf{x}) \in \{0, 1\}^J$, in which all values are equal to 0 except for the element in the position with the minimum output value for Eq. (24) that is equal to 1. In addition, the term $\|\boldsymbol{\beta}_j\|^2$ in the kernel version of the model without direct links is defined as:

$$\|\boldsymbol{\beta}_j\|^2 = \|\left(\mathbf{h}(\mathbf{x}_1)\ \dots\ \mathbf{h}(\mathbf{x}_N)\right)\boldsymbol{\alpha}\|^2 = \boldsymbol{\alpha}_j'(\Omega_{\mathrm{G}})\boldsymbol{\alpha}_j, \tag{25}$$

where $\Omega_{\mathrm{G}} = \begin{pmatrix} \mathbf{h}(\mathbf{x}_1) \\ \vdots \\ \mathbf{h}(\mathbf{x}_N) \end{pmatrix} (\mathbf{h}(\mathbf{x}_1)\ \dots\ \mathbf{h}(\mathbf{x}_N)) = \left(\Omega_{G_{i,j}}\right)_{i,j=1,\dots,N} \in \mathbb{R}^{N\times N}$ is the gaussian kernel matrix, which is defined as:

$$\Omega_{G_{i,j}} = k(\boldsymbol{x}_i, \boldsymbol{x}_j). \tag{26}$$

Accordingly, the term $\|\boldsymbol{\beta}_j\|^2$ in the kernel version of the model with direct links is defined as:

$$\|\boldsymbol{\beta}_j\|^2 = \boldsymbol{\alpha}_j'(\Omega_{\mathrm{G}} + \Omega_{\mathrm{L}})\boldsymbol{\alpha}_j, \tag{27}$$

where $\Omega_{\mathrm{L}} = \begin{pmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_N \end{pmatrix} (\mathbf{x}_1\ \dots\ \mathbf{x}_N) = \left(\Omega_{L_{i,j}}\right)_{i,j=1,\dots,N} \in \mathbb{R}^{N\times N}$ is the linear kernel matrix, which is defined as:

$$\Omega_{L_{i,j}} = l(\boldsymbol{x}_i, \boldsymbol{x}_j). \tag{28}$$

2.4. Analysis of the computational burden

In this section, the computational complexity of the PTRELM method is firstly analyzed and primarily compared with the base ELM neural network model. The comparison of the kernel methods is trivial as they compute a similar formulation to their neural networks comparison but scaling in N (number of training patterns) instead of in D (number of hidden nodes). The computational complexity of the ELM model (in its neural network version) is conditioned by the number of hidden nodes, D, and the size of the training set, N, as it must invert a matrix of dimension $D \times D$ and multiply the result with a matrix of dimension $D \times N$. The computational complexity of the multiplication of two matrices $D \times N$ and $D \times D$ is $O(D^2 \cdot N)$, and the complexity of inverting the matrix of dimension D is $O(D^3)$ [44]. Therefore, the computational complexity of the ELM method (in its neural network version) is $O(D^3 + (D^2 \cdot N))$.

The analysis for RVFL models is the same but taking into account that the complexity of the RVFL model (in its neural network version) depends on the number of hidden nodes, D, the number of attributes, K, and the size of the training set, N, as it must invert a matrix of dimension $(D+K)\times(D+K)$ and multiply the result with a matrix of dimension $(D+K)\times N$.

In the case of the PTRELM model, it has to multiply J times two matrices of dimensions $N_j \times D$ for the computation of the different $\mathbf{M}_j$ matrices, $O(J \cdot D^2 \cdot N_j)$, to invert J times matrices of dimensions $(D+1\times D+1)$, $O(J\cdot(D+1)^3)$, to multiply again two matrices of dimensions $(D+1\times D+1)$, $O(J\cdot(D+1)^3)$ and finally to solve J times an eigenvalue problem with matrices of dimensions again $(D+1\times D+1)$, $O(J\cdot(D+1)^3)$. In this way, the computational complexity of the proposed method is $O((J\cdot D^2\cdot N_j)+3\cdot(J\cdot(D+1)^3))$. Finally, it is important to remind that the PTRRVFL has a similar computational complexity but taking into account that the design matrices, $\mathbf{M}_j$, have, in this case, dimensions $(D+K)\times(D+K)$ and consequently, the computations increase with the factor K if compared to the PTRELM version of the model.

3. Experimental settings

This section summarizes the experimental study conducted to show the competitive performance of the proposed optimization model for RFNNs with and without direct links. Section 3.1 presents a description of the datasets used in the experimental validation. The measures employed for the performance evaluation are detailed in Section 3.2, whereas the experimental configuration of our proposal is given in Section 3.3. Finally, the implemented statistical tests to validate the results are included in Section 3.4.

3.1. Datasets

The experimental design has been set up to validate our proposal on classification problems. For this purpose, 22 datasets have been selected to test the robustness of the method when applied to very different problems. Thus, the datasets used include differences in the number of patterns, attributes, classes, and the number of patterns per class.

Table 1 summarizes the properties of the selected datasets, which have been extracted from the UCI repository [45]. It shows for each dataset, its name (*Dataset*), the number of patterns and attributes (*Size*), the number of classes (*#Classes*) and the distribution of instances within classes (*Class distribution*). It can be seen that both binary and multiclass problems have been chosen, varying the number of classes from 2 to 11. The *balloons* datasets have the minimum number of patterns (20), whereas *mushroom* has 8124 as the maximum of our datasets. The number of attributes ranges from 2 to 111. Furthermore, while several databases are balanced, others reach an imbalance ratio (the number of patterns of the majority class divided by the number of patterns of the minority class) of 7.33.

As the format of the UCI datasets is not always consistent, each one has been downloaded and processed in a standard format. It involves (1) removal of missing values by rows or by columns depending on which retains more information, (2) binarization of categorical input attributes, (3) scaling of input attributes in the interval [0, 1] due to their importance for distance-based classifiers such as ELM, and (4) labels have been binarized following a *1-to-J* encoding.

Given the stochasticity of the algorithms, a ten-fold cross-validation procedure (3 repetitions per fold) has been carried out. Therefore, 30 results are obtained for all compared algorithms, ensuring an adequate statistical significance of the results. The partitions are the same for all compared methods.

3.2. Measures

Two performance measures are used to evaluate the efficacy of the proposed method: correct classification rate and separability.

- Correct Classification Rate (*CCR*): is the proportion of successful hits (correct classifications) from all predictions made. It has been by far one of the most widely used metrics for assessing classifier performance for years. It is defined as:

$$CCR = \frac{1}{N}\sum_{i=1}^{N} I(\widehat{y}(\mathbf{x}_n) = \mathbf{y}_n), \tag{29}$$

where $\mathbf{y}_n \in \{0, 1\}^J$ and $\widehat{y}(\mathbf{x}_n) \in \{0, 1\}^J$ are the real and estimated target of pattern $\mathbf{x}_n$ according to the 1-of-J encoding, and $I(\cdot)$ is the zero–one loss function.

Table 1
Characteristics of alphabetically sorted datasets.

Dataset	Size	#Classes	Class distribution
Balance-scale	(625, 4)	3	(288, 288, 49)
Balloons-a	(20, 4)	2	(12, 8)
Balloons-b	(20, 4)	2	(12, 8)
Balloons-c	(20, 4)	2	(12, 8)
Breast-cancer	(286, 39)	2	(218, 68)
Breast-cancer-wisconsin	(683, 9)	2	(444, 239)
Breast-cancer-wisconsin-diagnostic	(569, 30)	2	(357, 212)
Congressional-voting-records	(435, 48)	2	(267, 168)
Connectionist-bench	(990, 10)	11	(90, ..., 90)
Dermatology	(358, 34)	6	(111, 71, 60, 48, 48, 20)
Fertility	(100, 9)	2	(88, 12)
Hill-valley	(606, 100)	2	(305, 301)
Iris	(150, 4)	3	(50, 50, 50)
Lung-cancer	(32, 146)	3	(9, 13, 10)
Monks-problems-1	(124, 6)	2	(62, 62)
Monks-problems-2	(432, 6)	2	(290, 142)
Mushroom	(8124, 111)	2	(4208, 3916)
Spect-heart	(80, 22)	2	(40, 40)
Thoracic-surgery	(470, 27)	2	(400, 70)
Thyroid-disease-new-thyroid	(215, 5)	3	(150, 35, 30)
Tic-tac-toe-endgame	(958, 27)	2	(626, 332)
Wall-following-robot-navigation	(5456, 2)	4	(2205, 2097, 826, 328)

- Separability (SEP): in this work, the separability between the predicted classes is also tested. SEP tests, for each class, how close the projections of the patterns of that class are to their reference point and how far away the patterns that do not belong to that class are with respect to this point. Mathematically:

$$SEP = \sum_{j=1}^{J} \left(\frac{\sum_{\mathbf{x}\in C_j}(\mathbf{h}'(\mathbf{x})\boldsymbol{\beta}_j - \beta_{0j})}{\sum_{\mathbf{x}\notin C_j}(\mathbf{h}'(\mathbf{x})\boldsymbol{\beta}_j - \beta_{0j})} \right), \tag{30}$$

  where $\mathbf{h}'(\mathbf{x})\boldsymbol{\beta}_j$ is the numerical output of the model for the jth class, and β_{0j} is the reference point of the j-class (which in the state-of-the-art ELM-based models is equal to one). As previously described, the output for the RVFL-base models for the jth class and the test pattern $\mathbf{x}$ is defined as $(\mathbf{h}'(\mathbf{x})\ \mathbf{x}')\boldsymbol{\beta}_j$.

3.3. Algorithms and configuration

The first three stages compare the RFNN models with and without direct links. For this purpose, pairwise comparisons of the algorithms are made using parametrized target outputs proposed in this work with respect to their standard counterparts models of the literature. Thus, the first phase consists of testing the performance of the new optimization on the neural versions of RFNNs (ELM and RVFL). The second stage works with their kernel versions (KRR and KRVFL). Furthermore, the third stage compares the implemented deep models (DeepELM and DeepRVFL). Finally, in the fourth phase, a qualitative analysis is carried out, showing graphically the projections made by each method to see the separability between classes intuitively.

The experimental configuration of our proposals, as well as that of the state-of-the-art algorithms, is described below:

- **ELM**: Extreme Learning Machine (described in Section 1).
- **PTRELM**: Parametrized Target outputs Regularized ELM (described in Section 2.1.4).
- **RVFL**: Random Vector Functional Link neural networks (described in Section 1).
- **PTRRVFL**: Parametrized Target outputs RVFL (described in Section 2.2).
- **KRR**: An implementation of the Kernel Ridge Regression model [46], which solves the least squares optimization with the L^2-norm regularization (ridge regression) for classification problems using the 1-of-J encoding and applying the kernel trick in its non-linear version. It is identical to the KELM [19], but the former will be used due to its original development.
- **KPTRELM**: Kernel version of PTRELM (described in Section 2.1.5).
- **KRVFL**: Kernel version of RVFL [47].
- **KPTRRVLF**: Kernel version of PTRRVFL (described in Section 2.2).
- **DeepELM**: An implementation of a deep model based on autoencoders for ELM [48].
- **DeepPTRELM**: Deep model in which the hidden layers are determined through autoencoders (as suggested in [48]), whereas the parameters of the output layer are estimated according to the proposed optimization described in Section 2.1.4.
- **DeepRVFL**: Adaptation of the deep model proposed in [48] to RVFL.
- **DeepPTRRVFL**: The deep version of PTRRVFL, where the parameters of the different hidden layers are computed through autoencoders [48] and the parameters of the output layer are determined with the proposed objective function and including direct links from the input to the output layer.

All non-kernel variants used a sigmoid activation function for a single layer of $D = 1000$ neurons. The regularization parameter, C, has been determined by a grid search through 5-fold nested cross-validation for all the models considered for comparison purposes ($C \in \{10^{-3}, \dots, 10^{3}\}$). In the case of kernel methods, the kernel function employed was the Gaussian one, and its corresponding parameter, σ, was also determined by a grid search ($\sigma \in \{10^{-3}, \dots, 10^{3}\}$). For deep models, the number of hidden layers was set to 4, as in [48]. Aiming to give each layer the same competitive advantage as the no-deep models, the number of neurons in each hidden layer was set to 1000.

3.4. Statistical tests

In order to select the best method based on its performance, statistical analysis of the results is necessary. As mentioned above, comparisons will be on a pairwise basis. Since the nature of the datasets used does not ensure normality, we will proceed with

Table 2
Comparison of ELM against PTRELM in terms of *CCR* and *SEP*. The results are expressed as their mean and standard deviation: $Mean_{SD}$.

Datasets–algorithms	*CCR*(↑)		*SEP*(↓)	
	ELM	PTRELM	ELM	PTRELM
Balance-scale	$0.9111_{0.0065}$	$\mathbf{0.9452_{0.0173}}$	$0.2552_{0.0247}$	$\mathbf{0.0263_{0.0157}}$
Balloons-a	$\mathbf{1.0000_{0.0000}}$	$\mathbf{1.0000_{0.0000}}$	$0.6725_{0.2499}$	$\mathbf{0.3861_{0.3651}}$
Balloons-b	$\mathbf{1.0000_{0.0000}}$	$\mathbf{1.0000_{0.0000}}$	$0.7660_{0.3646}$	$\mathbf{0.2117_{0.1845}}$
Balloons-c	$0.9167_{0.1800}$	$\mathbf{1.0000_{0.0000}}$	$0.8603_{0.6405}$	$\mathbf{0.3212_{0.2706}}$
Breast-cancer	$0.7516_{0.0290}$	$\mathbf{0.7579_{0.0172}}$	$1.8425_{0.2773}$	$\mathbf{0.4961_{0.3278}}$
Breast-cancer-wisconsin	$0.9652_{0.0145}$	$\mathbf{0.9683_{0.0131}}$	$\mathbf{0.0770_{0.0245}}$	$0.7776_{1.0914}$
Breast-cancer-wisconsin-diagnostic	$0.9656_{0.0203}$	$\mathbf{0.9688_{0.0135}}$	$0.1283_{0.0343}$	$\mathbf{0.0262_{0.0090}}$
Congressional-voting-records	$0.9428_{0.0264}$	$\mathbf{0.9476_{0.0131}}$	$0.1311_{0.0559}$	$\mathbf{0.0443_{0.0152}}$
Connectionist-bench	$\mathbf{0.9003_{0.0133}}$	$0.8591_{0.0330}$	$0.1961_{0.0090}$	$\mathbf{0.0189_{0.0036}}$
Dermatology	$\mathbf{0.9765_{0.0118}}$	$0.9672_{0.0151}$	$\mathbf{0.0966_{0.0367}}$	$0.6822_{0.5628}$
Fertility	$0.8788_{0.0000}$	$\mathbf{0.8976_{0.0351}}$	$6.8467_{0.3684}$	$\mathbf{4.8265_{1.8609}}$
Hill-valley	$0.7436_{0.0444}$	$\mathbf{0.7485_{0.0425}}$	$0.8579_{0.1226}$	$\mathbf{0.5576_{0.4312}}$
Iris	$0.9640_{0.0227}$	$\mathbf{0.9680_{0.0193}}$	$0.0805_{0.0268}$	$\mathbf{0.0113_{0.0043}}$
Lung-cancer	$0.4500_{0.0527}$	$\mathbf{0.5400_{0.1174}}$	$1.3920_{0.1012}$	$\mathbf{0.8437_{0.4367}}$
Monks-problems-1	$0.8528_{0.0334}$	$\mathbf{0.9382_{0.0246}}$	$0.3779_{0.0366}$	$\mathbf{0.1303_{0.0346}}$
Monks-problems-2	$0.6958_{0.0453}$	$\mathbf{0.8285_{0.0379}}$	$1.5676_{0.5674}$	$\mathbf{0.6950_{0.6195}}$
Mushroom	$0.9989_{0.0009}$	$\mathbf{1.0000_{0.0000}}$	$0.0063_{0.0049}$	$\mathbf{0.0004_{0.0002}}$
Spect-heart	$\mathbf{0.7308_{0.0811}}$	$0.7077_{0.1121}$	$\mathbf{1.3695_{0.6478}}$	$1.7928_{1.3433}$
Thoracic-surgery	$\mathbf{0.8385_{0.0291}}$	$0.7949_{0.0707}$	$4.7001_{0.6293}$	$\mathbf{4.5050_{1.4304}}$
Thyroid-disease-new-thyroid	$0.9127_{0.0374}$	$\mathbf{0.9592_{0.0315}}$	$0.1416_{0.0859}$	$\mathbf{0.0421_{0.0314}}$
Tic-tac-toe-endgame	$0.9806_{0.0087}$	$\mathbf{0.9856_{0.0066}}$	$0.0704_{0.0271}$	$\mathbf{0.0161_{0.0046}}$
Wall-following-robot-navigation-2	$0.7401_{0.0093}$	$\mathbf{0.7884_{0.0141}}$	$0.4055_{0.0097}$	$\mathbf{0.0499_{0.0017}}$
Mean values	0.8689	**0.8896**	1.0383	**0.7483**
Mean rankings ($\bar{R}$)	1.7727	**1.2273**	1.8636	**1.1364**

The best result is shown in **bold**.

non-parametric tests, as the assumptions for parametric tests are not met [49].

For comparisons between each pair of models, the Wilcoxon signed-rank test will be used [50]. It is a non-parametric statistical hypothesis test used to test the location of a population from a sample of data or to compare the locations of two populations using two paired samples.

Specifically, one-sided tests will be used, where the null hypothesis is $H_0 : M_{e1} = M_{e2}$, and the alternative hypothesis is $H_1 : M_{e1} > M_{e2}$ in the case of *CCR*, or $H_1 : M_{e1} < M_{e2}$ in the case of SEP, being M_{eX} the median of the model X.

4. Results

This section discusses the results obtained by the different methods when applied to the commented datasets. Therefore, the performance of the different methods, together with their statistical analysis of them, are included in the following subsections. As aforementioned, the experimental validation is divided into four phases.

4.1. Comparison in neural versions of RFNNs

All results tables show the comparison of the models two by two, i.e. the standard version versus the one proposed in the manuscript. The mean result and the standard deviation of the 30 runs are shown for each dataset and method. Note that standard deviation is indicated as a subscript. Furthermore, the mean values obtained by each method in each measure $\overline{CCR}$ and $\overline{SEP}$, as well as the mean rank $\bar{R}$, are also included at the end of the table. The best result for each dataset is highlighted in bold.

Table 2 shows the performance of the PTRELM model along with the baseline ELM model. As can be seen, PTRELM improves the standard ELM in 18 out of 22 datasets in terms of $\overline{CCR}$, which is reflected in the mean ranking obtained by them: $\bar{R} = 1.2273$ and $\bar{R} = 1.7727$, in favor of PTRELM. In addition, PTRELM achieves a $\overline{CCR}$ of 0.8896, whereas ELM gets 0.8689, showing a performance improvement. The results of the standard deviations are relatively low and similar between both algorithms, which means that the algorithms do not depend to a large extent on the initialization.

When analyzing the results in *SEP*, PTRELM achieves the best values (lowest errors) in 19 datasets, which translates into an average ranking of $\bar{R} = 1.1364$ compared to $\bar{R} = 1.8636$ obtained by the ELM algorithm. Besides, the differences in $\overline{SEP}$ obtained for all datasets are more noticeable: 0.7483 and 1.0383, respectively.

The Wilcoxon test reports a p-value of 0.0159 in the case of *CCR* and 0.0074 in the case of SEP. This means that in both situations, the null hypothesis is rejected since there are significant differences in favor of the proposed methodology under a level of significance $\alpha = 0.05$ (p-value $< \alpha$).

Table 3 shows the comparative information when using RFNNs with direct links, i.e. RVFL. In this case, the results are quite similar to the ELM case. On the one hand, the *CCR* of the proposed PTRRVFL methodology is better in 18 out of 22 databases, which means a value of 0.8882 compared to 0.8751 for RVFL. This can also be seen in the average ranking obtained by each model, being 1.25 versus 1.75.

In the same vein, the SEP is improved by this paper's proposal on 19 databases, with an average ranking of 1.1364 versus 1.8636. In addition, the differences in $\overline{SEP}$ are also more noticeable, being 0.7406 obtained by the PTRRVFL with respect to 1.0265 of the RVFL.

The Wilcoxon test rejects the null hypothesis that both algorithms perform equally. In this sense, for both *CCR* and *SEP*, our methodology is statistically better at the 0.05 significance level. The p-value in the case of CCR is equal to 0.0436, and in the case of SEP, it is 0.0106.

Finally, we have also implemented in a MATLAB code framework the ELM-GA, RVFL-GA, ELM-PSO, and RVFL-PSO (ELM and RVFL models in which the parameters associated with the hidden layer are not randomly determined but computed through a Genetic Algorithm, GA and Particle Swarm Optimization, PSO) and compared their results with their corresponding counterparts proposed in the manuscripts. The results are statistically in line with the reported in the randomized baseline models, so tables for those comparisons are not included in the manuscript (aiming not to overwhelm the readers with uninformative Tables).

Table 3
Comparison of RVFL against PTRRVFL in terms of CCR and SEP. The results are expressed as their mean and standard deviation: $Mean_{SD}$.

Datasets–algorithms	$CCR(\uparrow)$		$SEP(\downarrow)$	
	RVFL	PTRRVFL	RVFL	PTRRVFL
Balance-scale	$0.9111_{0.0065}$	$\mathbf{0.9471_{0.0174}}$	$0.2543_{0.0248}$	$\mathbf{0.0205_{0.0041}}$
Balloons-a	$\mathbf{1.0000_{0.0000}}$	$\mathbf{1.0000_{0.0000}}$	$0.6996_{0.2408}$	$\mathbf{0.3849_{0.3627}}$
Balloons-b	$\mathbf{1.0000_{0.0000}}$	$\mathbf{1.0000_{0.0000}}$	$0.7451_{0.3511}$	$\mathbf{0.2127_{0.1851}}$
Balloons-c	$0.9500_{0.1581}$	$\mathbf{1.0000_{0.0000}}$	$0.7183_{0.4104}$	$\mathbf{0.3221_{0.2701}}$
Breast-cancer	$0.7547_{0.0290}$	$\mathbf{0.7600_{0.0184}}$	$1.8058_{0.2713}$	$\mathbf{0.5295_{0.3145}}$
Breast-cancer-wisconsin	$0.9652_{0.0145}$	$\mathbf{0.9674_{0.0136}}$	$\mathbf{0.0771_{0.0247}}$	$1.0275_{1.1337}$
Breast-cancer-wisconsin-diagnostic	$0.9656_{0.0194}$	$\mathbf{0.9709_{0.0125}}$	$0.1373_{0.0334}$	$\mathbf{0.0322_{0.0256}}$
Congressional-voting-records	$0.9366_{0.0249}$	$\mathbf{0.9462_{0.0141}}$	$0.1397_{0.0531}$	$\mathbf{0.0447_{0.0151}}$
Connectionist-bench	$\mathbf{0.9036_{0.0132}}$	$0.8591_{0.0313}$	$0.1939_{0.0088}$	$\mathbf{0.0189_{0.0036}}$
Dermatology	$\mathbf{0.9773_{0.0119}}$	$0.9647_{0.0147}$	$\mathbf{0.0946_{0.0353}}$	$0.7794_{0.5569}$
Fertility	$0.8788_{0.0000}$	$\mathbf{0.8806_{0.0356}}$	$6.8197_{0.3957}$	$\mathbf{4.1580_{1.2756}}$
Hill-valley	$0.7426_{0.0438}$	$\mathbf{0.7559_{0.0407}}$	$0.8687_{0.1203}$	$\mathbf{0.4645_{0.2061}}$
Iris	$0.9600_{0.0249}$	$\mathbf{0.9680_{0.0193}}$	$0.0842_{0.0274}$	$\mathbf{0.0114_{0.0044}}$
Lung-cancer	$\mathbf{0.5200_{0.0789}}$	$\mathbf{0.5200_{0.1033}}$	$1.4092_{0.0374}$	$\mathbf{0.8137_{0.4649}}$
Monks-problems-1	$0.8556_{0.0308}$	$\mathbf{0.9438_{0.0213}}$	$0.3776_{0.0370}$	$\mathbf{0.1160_{0.0325}}$
Monks-problems-2	$0.6993_{0.0455}$	$\mathbf{0.8417_{0.0446}}$	$1.4759_{0.4667}$	$\mathbf{0.6922_{0.6179}}$
Mushroom	$0.9989_{0.0008}$	$\mathbf{1.0000_{0.0000}}$	$0.0064_{0.0049}$	$\mathbf{0.0003_{0.0002}}$
Spect-heart	$\mathbf{0.7423_{0.0603}}$	$0.7077_{0.0910}$	$\mathbf{1.2155_{0.3508}}$	$1.8229_{1.3500}$
Thoracic-surgery	$\mathbf{0.8513_{0.0041}}$	$0.7872_{0.0731}$	$4.8602_{0.4618}$	$\mathbf{4.7421_{1.9906}}$
Thyroid-disease-new-thyroid	$0.9127_{0.0374}$	$\mathbf{0.9507_{0.0347}}$	$0.1380_{0.0848}$	$\mathbf{0.0349_{0.0268}}$
Tic-tac-toe-endgame	$0.9834_{0.0061}$	$\mathbf{0.9856_{0.0066}}$	$0.0596_{0.0207}$	$\mathbf{0.0145_{0.0043}}$
Wall-following-robot-navigation-2	$0.7429_{0.0095}$	$\mathbf{0.7847_{0.0127}}$	$0.4012_{0.0087}$	$\mathbf{0.0498_{0.0017}}$
Mean values	0.8751	**0.8882**	1.0265	**0.7406**
Mean rankings ($\bar{R}$)	1.7500	**1.2500**	1.8636	**1.1364**

The best result is shown in **bold**.

4.2. Comparison in kernel versions of RFNNs

Table 4 includes the results of the kernel version comparison in the case of the ELM. In this case, the CCR results indicate that both algorithms have a similar ranking performance (both have an $\bar{R} = 1.5$ due to both being the best ones in 12 datasets). Still, the proposed methodology is slightly lower on average, with values of 0.8623 and 0.8821, respectively.

Nonetheless, the proposed methodology has a much more considerable outperformance in the case of the *SEP* metric. This translates to an improvement on $\overline{SEP}$ from 1.4055 to 1.0957. In addition, the average ranking of the KPTRELM algorithm is much more significant than in the case of the *CCR*, with a value of 1.2273 versus 1.7727 obtained by the KRR.

In the case of *CCR*, the Wilcoxon test returns a *p*-value of 0.18019 accepting the null hypothesis which states that both algorithms performs equally for a level of significance $\alpha = 0.05$ and $\alpha = 0.10$ (note that $p > \alpha$ in both cases). However, when the test is applied to results in *SEP*, the *p*-value is 0.0321, which means that the differences are statistically significant with $\alpha = 0.05$. In conclusion, it can be said that considering both metrics, our proposal improves statistically on the standard model.

Table 5 presents the results when comparing KPTRRVFL against KRVFL. Here the results are pretty striking. On the one hand, the *CCR* indicates that the best algorithm is the standard KRVFL version, with an average value $\overline{CCR}$ of 0.8729 versus 0.8623. In addition, the mean ranking is 1.3182 versus 1.6818.

On the other hand, in the *SEP* metric, the ranking results are opposite since KPTRRVFL obtains an average ranking $\bar{R}$ of 1.3182 versus 1.6818 obtained by KRVFL. Furthermore, in $\overline{SEP}$, the proposal slightly improves the standard version (0.9794 and 0.9925, respectively).

In the case of *CCR*, the Wilcoxon test returns a *p* value of 0.1148 accepting the null hypothesis which states that both models performs equally for a level of significance $\alpha = 0.05$ and $\alpha = 0.10$ (note that $p > \alpha$ in both cases). Similarly, when the test is applied to results in *SEP*, the *p* value is 0.2687, which means that the differences are not statistically significant. It can be concluded that there is a tie between the two methodologies.

4.3. Comparison in deep models of RFNNs

The comparison results between DeepELM and DeepPTRELM are shown in Table 6. Generally, the results obtained for this methodology are worse than the rest. It is because the architecture of the deep model has not been trained since the purpose of the comparative is to know whether the proposed optimization positively influences the final results with the same architecture. Having said that, we must say that it is the only case where our model is worse in terms of *CCR* and *SEP*.

Thus, $\overline{CCR}$ of DeepELM outperforms the $\overline{CCR}$ of DeepPTRELM with values of 0.7376 and 0.6867, respectively. The mean rankings $\bar{R}$ are not very different (1.4318 and 1.5682). In *SEP*, the differences are larger in favor of DeepELM, with $\bar{R}$ equal to 1.1818 and 1.8182. The Wilcoxon tests corroborate those results. *CCR* and *SEP* *p*-values are 0.0238 and 0.0012, respectively. The tests state that there are statistical differences with $\alpha = 0.05$ in favor of DeepELM.

Finally, results of DeepRVFL and DeepPTRRVFL are compared in Table 7. In this case, it is remarkable how the version with direct links achieves outstanding results without the need to optimize the architecture, as in the case of DeepELM.

Regarding *CCR*, the DeepPTRRVFL proposal improves the deep standard model in 15 databases. In addition, the average ranking is 1.3636 versus 1.6364. The results in $\overline{CCR}$ also affirm that the proposal improves with a value of 0.8453 versus 0.8244. In SEP, the results are also better in 14 of the 22 databases. This can be seen summarized in the values of $\bar{R}$ (1.3182 and 1.6818), and in the values of $\overline{SEP}$ (0.8670 versus 1.2469).

Wilcoxon tests determine the existence of significant differences in favor of our DeepPTRRVFL proposal in both metrics with a significance value of $\alpha = 0.10$. Here, the *p*-values are 0.0925 and 0.0635 (*p*-values $< \alpha$ in both cases).

Table 4
Comparison of KRR against KPTRELM in terms of CCR and SEP. The results are expressed as their mean and standard deviation: $Mean_{SD}$.

Datasets–algorithms	$CCR(\uparrow)$		$SEP(\downarrow)$	
	KRR	KPTRELM	KRR	KPTRELM
Balance-scale	$0.8990_{0.0128}$	$\mathbf{0.9476_{0.0256}}$	$0.2019_{0.0185}$	$\mathbf{0.1362_{0.4185}}$
Balloons-a	$\mathbf{0.9667_{0.1054}}$	$\mathbf{0.9667_{0.1054}}$	$2.0500_{0.1581}$	$\mathbf{1.2118_{0.5222}}$
Balloons-b	$\mathbf{1.0000_{0.0000}}$	$0.9167_{0.1416}$	$1.5635_{0.8349}$	$\mathbf{0.9235_{0.9110}}$
Balloons-c	$\mathbf{0.9667_{0.0703}}$	$0.9333_{0.1405}$	$1.8709_{0.6042}$	$\mathbf{0.9530_{0.7333}}$
Breast-cancer	$0.7284_{0.0396}$	$\mathbf{0.7589_{0.0213}}$	$\mathbf{2.3772_{0.7173}}$	$2.9506_{0.9640}$
Breast-cancer-wisconsin	$0.9648_{0.0145}$	$\mathbf{0.9670_{0.0130}}$	$0.7805_{0.8631}$	$\mathbf{0.0701_{0.0932}}$
Breast-cancer-wisconsin-diagnostic	$\mathbf{0.9704_{0.0128}}$	$0.9608_{0.0142}$	$0.8206_{0.7383}$	$\mathbf{0.0185_{0.0145}}$
Congressional-voting-records	$\mathbf{0.9524_{0.0214}}$	$0.9207_{0.0176}$	$0.1353_{0.0922}$	$\mathbf{0.0458_{0.0212}}$
Connectionist-bench	$\mathbf{0.9782_{0.0123}}$	$0.9306_{0.0296}$	$0.6063_{0.5239}$	$\mathbf{0.2424_{0.0335}}$
Dermatology	$\mathbf{0.9647_{0.0213}}$	$0.9025_{0.0542}$	$0.4493_{0.5655}$	$\mathbf{0.0459_{0.0177}}$
Fertility	$0.8697_{0.0379}$	$\mathbf{0.8758_{0.0096}}$	$\mathbf{4.9866_{2.5573}}$	$5.7158_{4.9468}$
Hill-valley	$\mathbf{0.6743_{0.0348}}$	$0.4975_{0.0249}$	$\mathbf{1.1355_{0.1070}}$	$3.1870_{2.1648}$
Iris	$0.9400_{0.0411}$	$\mathbf{0.9420_{0.0290}}$	$1.1023_{0.5525}$	$\mathbf{0.0084_{0.0073}}$
Lung-cancer	$0.3800_{0.1317}$	$\mathbf{0.5300_{0.0949}}$	$1.4433_{0.1146}$	$\mathbf{0.9934_{0.2940}}$
Monks-problems-1	$0.8792_{0.0376}$	$\mathbf{0.9347_{0.0286}}$	$0.3517_{0.0585}$	$\mathbf{0.1112_{0.0389}}$
Monks-problems-2	$0.7576_{0.0421}$	$\mathbf{0.8118_{0.0226}}$	$1.9582_{0.7578}$	$\mathbf{0.1867_{0.1031}}$
Mushroom	$\mathbf{0.9998_{0.0004}}$	$0.9412_{0.0043}$	$2.0052_{0.0000}$	$\mathbf{0.0030_{0.0009}}$
Spect-heart	$\mathbf{0.7346_{0.0665}}$	$\mathbf{0.7346_{0.0757}}$	$1.2411_{0.3311}$	$\mathbf{0.9465_{0.5933}}$
Thoracic-surgery	$0.8429_{0.0172}$	$\mathbf{0.8526_{0.0000}}$	$\mathbf{4.2290_{0.5041}}$	$5.9417_{0.0173}$
Thyroid-disease-new-thyroid	$0.9690_{0.0197}$	$\mathbf{0.9704_{0.0261}}$	$1.4085_{1.3602}$	$\mathbf{0.0190_{0.0188}}$
Tic-tac-toe-endgame	$\mathbf{0.9856_{0.0066}}$	$0.7404_{0.0197}$	$\mathbf{0.1078_{0.0110}}$	$0.3946_{0.0502}$
Wall-following-robot-navigation-2	$\mathbf{0.9827_{0.0025}}$	$0.9344_{0.0082}$	$0.0961_{0.2223}$	$\mathbf{0.0004_{0.0005}}$
Mean values	**0.8821**	0.8623	1.4055	**1.0957**
Mean rankings ($\bar{R}$)	**1.5000**	**1.5000**	1.7727	**1.2273**

The best result is shown in **bold**.

Table 5
Comparison of KRVFL against KPTRRVFL in terms of CCR and SEP. The results are expressed as their mean and standard deviation: $Mean_{SD}$.

Datasets–algorithms	$CCR(\uparrow)$		$SEP(\downarrow)$	
	KRVFL	KPTRRVFL	KRVFL	KPTRRVFL
Balance-scale	$0.9010_{0.0151}$	$\mathbf{0.9029_{0.0326}}$	$\mathbf{0.2197_{0.0298}}$	$0.4421_{0.4780}$
Balloons-a	$0.9000_{0.1165}$	$\mathbf{1.0000_{0.0000}}$	$0.7667_{0.5905}$	$\mathbf{0.2175_{0.4344}}$
Balloons-b	$0.9500_{0.0805}$	$\mathbf{1.0000_{0.0000}}$	$\mathbf{0.7401_{0.4724}}$	$0.9822_{1.1941}$
Balloons-c	$0.7833_{0.1581}$	$\mathbf{1.0000_{0.0000}}$	$\mathbf{0.6837_{0.4775}}$	$1.0584_{2.5696}$
Breast-cancer	$\mathbf{0.7558_{0.0301}}$	$0.6884_{0.1480}$	$1.7814_{0.3846}$	$\mathbf{1.1218_{0.5720}}$
Breast-cancer-wisconsin	$\mathbf{0.9709_{0.0130}}$	$0.9665_{0.0174}$	$0.4148_{0.7458}$	$\mathbf{0.0870_{0.0380}}$
Breast-cancer-wisconsin-diagnostic	$\mathbf{0.9704_{0.0100}}$	$0.9487_{0.0146}$	$0.0781_{0.0180}$	$\mathbf{0.0210_{0.0127}}$
Congressional-voting-records	$\mathbf{0.9510_{0.0182}}$	$0.9345_{0.0196}$	$0.1104_{0.0406}$	$\mathbf{0.1058_{0.0335}}$
Connectionist-bench	$\mathbf{0.9818_{0.0091}}$	$0.9270_{0.0256}$	$0.0661_{0.0138}$	$\mathbf{0.0131_{0.0022}}$
Dermatology	$\mathbf{0.9723_{0.0164}}$	$0.8387_{0.0467}$	$0.0847_{0.0365}$	$\mathbf{0.0026_{0.0011}}$
Fertility	$\mathbf{0.8727_{0.0128}}$	$0.8364_{0.2330}$	$7.5644_{4.2977}$	$\mathbf{6.7023_{2.2154}}$
Hill-valley	$0.6799_{0.0287}$	$\mathbf{0.7178_{0.0692}}$	$\mathbf{1.3082_{0.2767}}$	$1.9276_{0.7541}$
Iris	$\mathbf{0.9600_{0.0211}}$	$0.9000_{0.0573}$	$0.0815_{0.0393}$	$\mathbf{0.0101_{0.0131}}$
Lung-cancer	$0.4400_{0.1075}$	$\mathbf{0.4800_{0.1549}}$	$1.3491_{0.0899}$	$\mathbf{0.7279_{0.5343}}$
Monks-problems-1	$\mathbf{0.8944_{0.0301}}$	$0.7722_{0.0424}$	$\mathbf{0.3292_{0.0451}}$	$0.5588_{0.1983}$
Monks-problems-2	$\mathbf{0.8111_{0.0224}}$	$0.7208_{0.1550}$	$0.6695_{0.0765}$	$\mathbf{0.6073_{0.9348}}$
Mushroom	$\mathbf{0.9988_{0.0010}}$	$0.9966_{0.0057}$	$0.0026_{0.0015}$	$\mathbf{0.0002_{0.0002}}$
Spect-heart	$0.6423_{0.0832}$	$\mathbf{0.8038_{0.1203}}$	$1.3826_{0.4180}$	$\mathbf{1.0329_{4.8332}}$
Thoracic-surgery	$\mathbf{0.8359_{0.0222}}$	$0.7596_{0.1360}$	$\mathbf{4.0364_{0.7385}}$	$4.9950_{1.8659}$
Thyroid-disease-new-thyroid	$\mathbf{0.9662_{0.0201}}$	$0.9493_{0.0313}$	$0.1039_{0.0445}$	$\mathbf{0.0492_{0.0304}}$
Tic-tac-toe-endgame	$\mathbf{0.9840_{0.0067}}$	$0.8997_{0.0378}$	$\mathbf{0.0401_{0.0083}}$	$0.8846_{0.1883}$
Wall-following-robot-navigation-2	$\mathbf{0.9828_{0.0021}}$	$0.9268_{0.0156}$	$0.0220_{0.0033}$	$\mathbf{0.0001_{0.0001}}$
Mean values	**0.8729**	0.8623	0.9925	**0.9794**
Mean rankings ($\bar{R}$)	**1.3182**	1.6818	1.6818	**1.3182**

The best result is shown in **bold**.

4.4. Qualitative analysis of the projections

This Section visually shows the numerical output of the model and the reference point for all classes obtained for the different algorithms. Thus, Figs. 4 and 5 illustrate the projections obtained for each pattern in the dataset *iris*. The y-axis of ordinates represents the classes, whereas the x-axis represents, for each jth class, its reference point (β_{0j}) and the value of the model ($f_j = \mathbf{v}'(\mathbf{x})\boldsymbol{\beta}_j$) for each test pattern ($\mathbf{x}$). The patterns belonging to each class are shown with a blue circle, while the patterns from other classes are marked with a cross.

Figs. 4 and 5 show that almost all methods accurately separate the first-class patterns. Nevertheless, classes two and three are more complicated, as seen in the overlapping of the blue and black dots. Despite this, with the exception of DeepPTRELM (see Fig. 5(d)), as discussed before, our optimization methodology manages to make this overlapping practically non-existent. A priori, although it may seem to exist, we are actually on a much

Table 6
Comparison of DeepELM against DeepPTRELM in terms of CCR and SEP. The results are expressed as their mean and standard deviation: $Mean_{SD}$.

Datasets–algorithms	$CCR(\uparrow)$		$SEP(\downarrow)$	
	DeepELM	DeepPTRELM	DeepELM	DeepPTRELM
Balance-scale	$0.7115_{0.0560}$	**$0.7154_{0.1204}$**	**$0.7408_{0.2253}$**	$1.7961_{0.0026}$
Balloons-a	$0.7000_{0.1054}$	**$0.7333_{0.2108}$**	**$2.1287_{0.7730}$**	$2.4999_{0.0002}$
Balloons-b	$0.6400_{0.1995}$	**$0.6500_{0.2284}$**	$5.6787_{10.8415}$	**$2.5000_{0.0001}$**
Balloons-c	**$0.6833_{0.0946}$**	$0.5667_{0.1405}$	$2.5364_{0.4103}$	**$2.5000_{0.0000}$**
Breast-cancer	**$0.7547_{0.0361}$**	$0.6337_{0.0539}$	**$1.9052_{0.8808}$**	$3.5008_{0.0819}$
Breast-cancer-wisconsin	**$0.9568_{0.0208}$**	$0.8749_{0.0886}$	**$0.1295_{0.0675}$**	$2.4072_{0.0002}$
Breast-cancer-wisconsin-diagnostic	**$0.9593_{0.0100}$**	$0.8349_{0.0493}$	**$0.1959_{0.0375}$**	$2.2882_{0.0001}$
Congressional-voting-records	**$0.9386_{0.0309}$**	$0.8462_{0.0423}$	**$0.1502_{0.0423}$**	$1.9890_{0.4834}$
Connectionist-bench	$0.5015_{0.0370}$	**$0.5376_{0.0258}$**	**$0.6624_{0.0240}$**	$1.1000_{0.0001}$
Dermatology	**$0.9429_{0.0259}$**	$0.7378_{0.0994}$	**$0.1380_{0.0226}$**	$1.2684_{0.0024}$
Fertility	**$0.8788_{0.0000}$**	$0.8888_{0.3487}$	**$7.3879_{0.0001}$**	**$7.3879_{0.0001}$**
Hill-valley	$0.4941_{0.0153}$	**$0.5030_{0.0138}$**	$2.0048_{0.0098}$	**$2.0001_{0.0003}$**
Iris	**$0.6720_{0.0215}$**	$0.6360_{0.1045}$	**$0.4497_{0.1021}$**	$1.5007_{0.0002}$
Lung-cancer	**$0.4700_{0.1160}$**	$0.4000_{0.1155}$	**$1.2901_{0.2447}$**	$1.5204_{0.0032}$
Monks-problems-1	**$0.5722_{0.0602}$**	$0.5438_{0.0714}$	**$1.6381_{0.3089}$**	$2.0000_{0.0001}$
Monks-problems-2	**$0.6688_{0.0034}$**	**$0.6688_{0.0034}$**	**$2.5145_{0.0234}$**	**$2.5145_{0.0234}$**
Mushroom	**$0.9842_{0.0044}$**	$0.8628_{0.0455}$	**$0.0611_{0.0052}$**	$1.4785_{0.7823}$
Spect-heart	$0.7077_{0.0413}$	**$0.7615_{0.1022}$**	**$1.2110_{0.5523}$**	$1.9997_{0.0002}$
Thoracic-surgery	**$0.8506_{0.0043}$**	$0.6397_{0.3002}$	**$5.5632_{0.8416}$**	$5.9556_{0.0000}$
Thyroid-disease-new-thyroid	$0.8239_{0.0489}$	**$0.9169_{0.1842}$**	**$0.8447_{0.6303}$**	$2.7283_{0.0001}$
Tic-tac-toe-endgame	$0.6918_{0.0284}$	**$0.7702_{0.1368}$**	**$1.3508_{0.1795}$**	$2.4207_{0.0091}$
Wall-following-robot-navigation-2	**$0.6245_{0.1217}$**	$0.3845_{0.1497}$	**$0.7992_{0.4043}$**	$1.5443_{0.0010}$
Mean values	**0.7376**	0.6867	**1.7900**	2.4955
Mean rankings ($\bar{R}$)	**1.4318**	1.5682	**1.1818**	1.8182

The best result is shown in **bold**.

Table 7
Comparison of DeepRVFL against DeepPTRRVFL in terms of CCR and SEP. The results are expressed as their mean and standard deviation: $Mean_{SD}$.

Datasets–algorithms	$CCR(\uparrow)$		$SEP(\downarrow)$	
	DeepRVFL	DeepPTRRVFL	DeepRVFL	DeepPTRRVFL
Balance-scale	$0.8885_{0.0119}$	**$0.8893_{0.0031}$**	$0.4247_{0.4169}$	**$0.0003_{0.0005}$**
Balloons-a	**$0.9500_{0.0805}$**	$0.9167_{0.1405}$	**$0.6620_{0.3374}$**	$0.6676_{0.6783}$
Balloons-b	$0.9333_{0.1405}$	**$1.0000_{0.0000}$**	**$0.9106_{0.4604}$**	$1.5079_{0.7601}$
Balloons-c	**$0.8667_{0.1892}$**	$0.8333_{0.0972}$	$0.7600_{0.6551}$	**$0.6925_{0.4358}$**
Breast-cancer	**$0.7537_{0.0273}$**	$0.7479_{0.0627}$	$2.0552_{0.6448}$	**$0.4924_{0.4474}$**
Breast-cancer-wisconsin	$0.9652_{0.0179}$	**$0.9687_{0.0155}$**	$0.0901_{0.0375}$	**$0.0676_{0.0989}$**
Breast-cancer-wisconsin-diagnostic	$0.9556_{0.0160}$	**$0.9698_{0.0067}$**	$0.1477_{0.0177}$	**$0.0265_{0.0089}$**
Congressional-voting-records	$0.9462_{0.0266}$	**$0.9469_{0.1751}$**	**$0.1265_{0.0483}$**	$0.3967_{0.4325}$
Connectionist-bench	$0.5015_{0.0370}$	**$0.5376_{0.0132}$**	**$0.6624_{0.0240}$**	$1.4327_{1.0145}$
Dermatology	**$0.9739_{0.0156}$**	$0.9605_{0.0218}$	$0.1178_{0.0321}$	**$0.1198_{0.2714}$**
Fertility	**$0.8788_{0.0202}$**	**$0.8788_{0.0192}$**	**$7.3444_{0.0511}$**	$6.8538_{0.0235}$
Hill-valley	**$0.6827_{0.0344}$**	$0.6653_{0.0220}$	$1.3415_{0.1920}$	**$0.0183_{0.0042}$**
Iris	$0.9160_{0.0858}$	**$0.9620_{0.0596}$**	**$0.2299_{0.3322}$**	$1.2769_{0.5481}$
Lung-cancer	$0.4400_{0.1075}$	**$0.5500_{0.0154}$**	$1.3606_{0.1212}$	**$0.6968_{0.5981}$**
Monks-problems-1	$0.6611_{0.0411}$	**$0.8014_{0.0915}$**	$1.1699_{0.4409}$	**$0.8538_{2.6128}$**
Monks-problems-2	$0.6639_{0.0082}$	**$0.6746_{0.0126}$**	$2.5174_{0.0580}$	**$1.4363_{0.2098}$**
Mushroom	$0.9983_{0.0014}$	**$1.0000_{0.0000}$**	**$0.0043_{0.0046}$**	$0.7275_{0.3417}$
Spect-heart	**$0.7385_{0.0698}$**	$0.7231_{0.0846}$	$1.2993_{0.5414}$	**$1.0270_{0.5548}$**
Thoracic-surgery	**$0.8442_{0.0176}$**	$0.7981_{0.2079}$	$5.3789_{0.9026}$	**$0.5888_{0.2510}$**
Thyroid-disease-new-thyroid	$0.8901_{0.0454}$	**$0.9577_{0.0248}$**	$0.2551_{0.0932}$	**$0.0643_{0.0275}$**
Tic-tac-toe-endgame	$0.9840_{0.0067}$	**$0.9840_{0.0087}$**	**$0.0400_{0.0083}$**	$0.0816_{0.0184}$
Wall-following-robot-navigation-2	$0.7047_{0.0611}$	**$0.8309_{0.0182}$**	$0.5338_{0.1669}$	**$0.0511_{0.0013}$**
Mean values	0.8244	**0.8453**	1.2469	**0.8670**
Mean rankings ($\bar{R}$)	1.6364	**1.3636**	1.6818	**1.3182**

The best result is shown in **bold**.

larger scale of values than the other methods, so the separability is much better. In this way, if we zoom in on these areas, the distance between the points of one class and another would be larger than those achieved by the other methods.

These results can be corroborated by what was stated analytically: our method correctly classifies the patterns of each class and separates the patterns of one class from the rest, which leads to better results in generalization in almost all models.

5. Conclusions and future research

This paper proposes an alternative classification model for Randomized-based Feedforward Neural Networks using direct

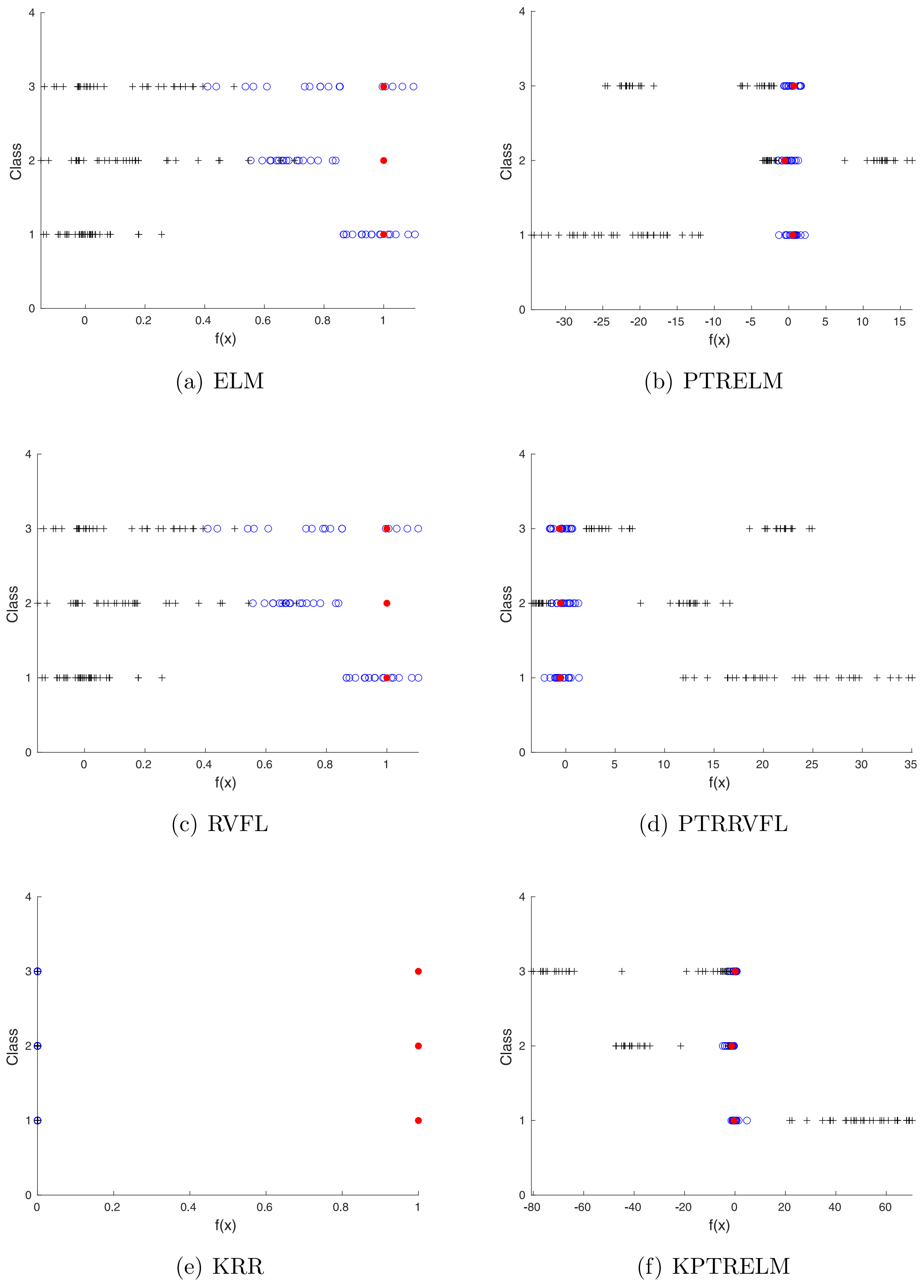

Fig. 4. Projections of test patterns for the iris dataset obtained by ELM, PTRELM, RVFL, PTRRVFL, KRR and KPTRELM. Patterns belonging to the class are shown by a blue circle, while patterns from other classes are denoted by a cross. In addition, the reference point of each class is included as a red filled circle.

links (RVFL) and without using it (ELM). In the proposed classification framework, the patterns per class are projected to a reference point (also tuned during the optimization procedure), and the remaining patterns are projected as far away as possible from that reference point. The optimization framework has been instantiated in two different versions: the neural network implementation and the kernelized version of the algorithm. The two algorithms are proposed in order to overcome the traditional drawback of the classification solution of the ELM framework of projecting the patterns of each class arbitrarily to one and the others to zero (being arbitrary not only the reference points but also the distance between the patterns belonging to the class and the remaining ones). Deep models have also been adapted to test the methodology's performance on them.

The performance of the methodology has been compared by making 2-to −2 pairwise comparisons. Each model has been compared without and using the new way of optimizing the models. In this way, the proposal's effect on different alternatives can be observed. In this comparison, 22 classification databases have been used regarding two classification metrics: the correct

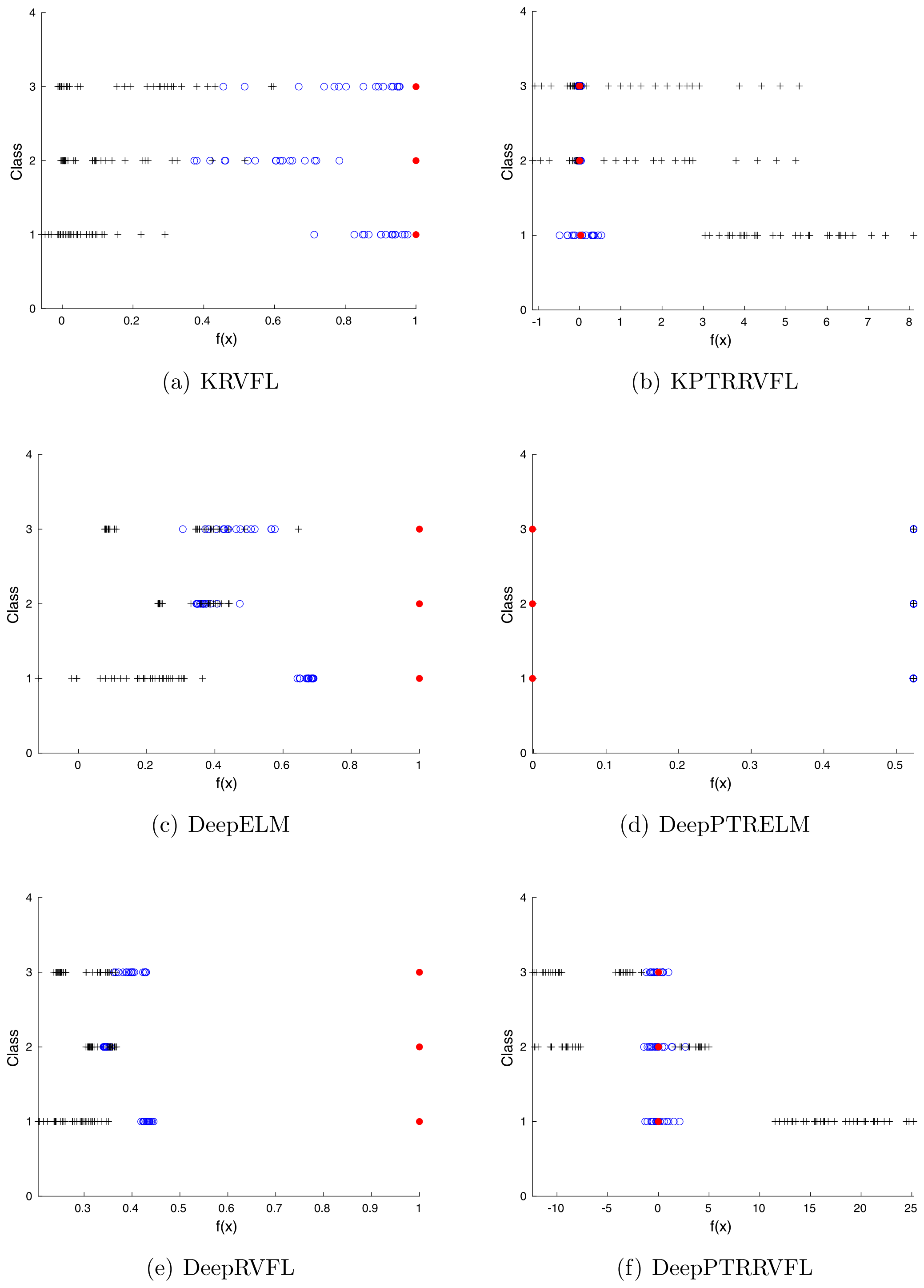

Fig. 5. Projections of test patterns for the iris dataset obtained by KRVFL, KPTRRVFL, DeepELM, DeepPTRELM, DeepRVFL and DeepPTRRVFL. Patterns belonging to the class are shown by a blue circle, while patterns from other classes are denoted by a cross. In addition, the reference point of each class is included as a red filled circle.

classification rate (CCR) and an index measuring the separability degree between the patterns belonging to the class and the remaining ones (SEP). The proposed methodology achieved competitive performance results in both CCR and SEP. Specifically, the proposal has improved four of the six configurations tested in this work, tied in one and worsened in another. Thus, we can humbly claim that the estimation of the reference point, and the way the optimization problem is presented, lead us to such desirable results. Additionally, the competitive performance of the model has also been shown qualitatively by plotting different projections of patterns for different classification problems (with different characteristics).

In future work, the L^2-norm of the optimization problem could be replaced with other choices to promote sparsity in the algorithm. The main problem of this approach is the increment of the computational burden of the proposal (as the algorithm cannot be analytically determined but iteratively). Furthermore, the authors plan to study deep architectures in more detail. Additionally, the authors also plan to incorporate the error function of the algorithm to adapt the optimization framework to ordinal regression problems and multi-label learning.

Funding

Funding for open access charge: Universidad de Málaga / CBUA

CRediT authorship contribution statement

Antonio Manuel Durán-Rosal: Methodology, Validation, Formal analysis, Investigation, Writing – original draft, Writing – review & editing, Visualization. **Aggeo Durán-Fernández:** Methodology, Software, Validation, Investigation, Data curation, Writing – original draft, Writing – review & editing. **Francisco Fernández-Navarro:** Software, Investigation, Validation, Conceptualization, Writing – original draft, Writing – review & editing. **Mariano Carbonero-Ruz:** Software, Investigation, Validation, Resources, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

References

[1] A.-L. Schubert, D. Hagemann, A. Voss, K. Bergmann, Evaluating the model fit of diffusion models with the root mean square error of approximation, J. Math. Psych. 77 (2017) 29–45.

[2] P. Gastaldo, R. Zunino, E. Cambria, S. Decherchi, Combining ELM with random projections, IEEE Intell. Syst. 28 (6) (2013) 46–48.

[3] C. Perales-González, F. Fernández-Navarro, J. Pérez-Rodríguez, M. Carbonero-Ruz, Negative correlation hidden layer for the extreme learning machine, Appl. Soft Comput. 109 (2021) 107482.

[4] J. Sánchez-Monedero, P.A. Gutiérrez, F. Fernández-Navarro, C. Hervás-Martínez, Weighting efficient accuracy and minimum sensitivity for evolving multi-class classifiers, Neural Process. Lett. 34 (2) (2011) 101–116.

[5] D. Chyzhyk, A. Savio, M. Graña, Evolutionary ELM wrapper feature selection for Alzheimer's disease CAD on anatomical brain MRI, Neurocomputing 128 (2014) 73–80.

[6] E.M. Figueiredo, T.B. Ludermir, Investigating the use of alternative topologies on performance of the PSO-ELM, Neurocomputing 127 (2014) 4–12, Advances in Intelligent Systems.

[7] B. Lacruz, D. Lahoz, P. Mateo, μG2-ELM: An upgraded implementation of μ G-ELM, Neurocomputing 171 (2016) 1302–1312.

[8] L. Zhang, P.N. Suganthan, Visual tracking with convolutional random vector functional link network, IEEE Trans. Cybern. 47 (10) (2016) 3243–3253.

[9] C. Perales-González, M. Carbonero-Ruz, J. Pérez-Rodríguez, D. Becerra-Alonso, F. Fernández-Navarro, Negative correlation learning in the extreme learning machine framework, Neural Comput. Appl. 32 (17) (2020) 13805–13823.

[10] B.B. Hazarika, D. Gupta, Modelling and forecasting of COVID-19 spread using wavelet-coupled random vector functional link networks, Appl. Soft Comput. 96 (2020) 106626.

[11] C. Perales-González, F. Fernández-Navarro, J. Pérez-Rodríguez, M. Carbonero-Ruz, Negative correlation hidden layer for the extreme learning machine, Appl. Soft Comput. 109 (2021) 107482.

[12] G.A. Kale, C. Karakuzu, Multilayer extreme learning machines and their modeling performance on dynamical systems, Appl. Soft Comput. 122 (2022) 108861.

[13] H. Tian, B. Meng, A new modeling method based on bagging ELM for day-ahead electricity price prediction, in: 5th International Conference on Bio-Inspired Computing: Theories and Applications, IEEE, 2010, pp. 1076–1079.

[14] R. Gao, L. Du, K.F. Yuen, P.N. Suganthan, Walk-forward empirical wavelet random vector functional link for time series forecasting, Appl. Soft Comput. 108 (2021) 107450.

[15] S. Scardapane, D. Wang, M. Panella, A. Uncini, Distributed learning for random vector functional-link networks, Inform. Sci. 301 (2015) 271–284.

[16] A. Gaspar, D. Oliva, S. Hinojosa, I. Aranguren, D. Zaldivar, An optimized kernel extreme learning machine for the classification of the autism spectrum disorder by using gaze tracking images, Appl. Soft Comput. 120 (2022) 108654.

[17] X. Xu, J. Deng, E. Coutinho, C. Wu, L. Zhao, B.W. Schuller, Connecting subspace learning and extreme learning machine in speech emotion recognition, IEEE Trans. Multimed. 21 (2019) 795–808.

[18] G.-B. Huang, Q.-Y. Zhu, C.-K. Siew, Extreme learning machine: theory and applications, Neurocomputing 70 (1–3) (2006) 489–501.

[19] G.-B. Huang, H. Zhou, X. Ding, R. Zhang, Extreme learning machine for regression and multiclass classification, IEEE Trans. Syst. Man Cybern. B 42 (2) (2011) 513–529.

[20] D. Husmeier, Random vector functional link (RVFL) networks, in: Neural Networks for Conditional Probability Estimation, Springer, 1999, pp. 87–97.

[21] Y. Ren, P.N. Suganthan, N. Srikanth, G. Amaratunga, Random vector functional link network for short-term electricity load demand forecasting, Inform. Sci. 367 (2016) 1078–1093.

[22] Q. Shi, R. Katuwal, P.N. Suganthan, M. Tanveer, Random vector functional link neural network based ensemble deep learning, Pattern Recognit. 117 (2021) 107978.

[23] A.N. Tikhonov, V.I. Arsenin, Solutions of Ill-Posed Problems, Vol. 14, Vh Winston, 1977.

[24] C. Perales-González, M. Carbonero-Ruz, D. Becerra-Alonso, J. Pérez-Rodríguez, F. Fernández-Navarro, Regularized ensemble neural networks models in the extreme learning machine framework, Neurocomputing 361 (2019) 196–211.

[25] Y. Zhang, Q. Wu, J. Hu, An adaptive learning algorithm for regularized extreme learning machine, IEEE Access 9 (2021) 20736–20745.

[26] Y. Wang, D. Li, Y. Du, Z. Pan, Anomaly detection in traffic using L1-norm minimization extreme learning machine, Neurocomputing 149 (2015) 415–425.

[27] X. Shi, Q. Kang, J. An, M. Zhou, Novel L1 regularized extreme learning machine for soft-sensing of an industrial process, IEEE Trans. Ind. Inform. 18 (2) (2021) 1009–1017.

[28] X. Luo, X. Chang, X. Ban, Regression and classification using extreme learning machine based on L1-norm and L2-norm, Neurocomputing 174 (2016) 179–186.

[29] Z. Zhou, J. Guo, Y. Wang, Z. Zhu, Random vector functional link network with L21 norm regularization for robot visual servo control with feature constraint, J. Mech. Sci. Technol. (2022) 1–13.

[30] L. Yang, S. Yang, S. Li, Z. Liu, L. Jiao, Incremental laplacian regularization extreme learning machine for online learning, Appl. Soft Comput. 59 (2017) 546–555.

[31] R. Hu, E. Ratner, D. Stewart, K.-M. Björk, A. Lendasse, A modified Lanczos Algorithm for fast regularization of extreme learning machines, Neurocomputing 414 (2020) 172–181.

[32] H. Ye, F. Cao, D. Wang, A hybrid regularization approach for random vector functional-link networks, Expert Syst. Appl. 140 (2020) 112912.

[33] O.L. Mangasarian, E.W. Wild, Multisurface proximal support vector machine classification via generalized eigenvalues, IEEE Trans. Pattern Anal. Mach. Intell. 28 (1) (2005) 69–74.

[34] R. Khemchandani, S. Chandra, et al., Twin support vector machines for pattern classification, IEEE Trans. Pattern Anal. Mach. Intell. 29 (5) (2007) 905–910.

[35] Q. Ye, N. Ye, Improved proximal support vector machine via generalized eigenvalues, in: 2009 International Joint Conference on Computational Sciences and Optimization, Vol. 1, IEEE, 2009, pp. 705–709.

[36] Y.-H. Shao, N.-Y. Deng, Z.-M. Yang, Least squares recursive projection twin support vector machine for classification, Pattern Recognit. 45 (6) (2012) 2299–2307.

[37] Y.-H. Shao, N.-Y. Deng, W.-J. Chen, Z. Wang, Improved generalized eigenvalue proximal support vector machine, IEEE Signal Process. Lett. 20 (3) (2012) 213–216.

[38] S. Sun, X. Xie, C. Dong, Multiview learning with generalized eigenvalue proximal support vector machines, IEEE Trans. Cybern. 49 (2) (2018) 688–697.

[39] C. Geng, S. Chen, Multiplane convex proximal support vector machine, IEEE Trans. Neural Netw. Learn. Syst. (2021).

[40] M. Tanveer, T. Rajani, R. Rastogi, Y. Shao, M. Ganaie, Comprehensive review on twin support vector machines, Ann. Oper. Res. (2022) 1–46.

[41] J.-G. Sun, Eigenvalues of Rayleigh quotient matrices, Numer. Math. 59 (1) (1991) 603–614.
[42] S. Boyd, S.P. Boyd, L. Vandenberghe, Convex Optimization, Cambridge University Press, 2004.
[43] Y. Xu, Q. Ye, Generalized Mercer Kernels and Reproducing Kernel Banach Spaces, Vol. 258, (1243) American Mathematical Society, 2019.
[44] S.A. Cook, An overview of computational complexity, Commun. ACM 26 (6) (1983) 400–408.
[45] D. Dua, C. Graff, UCI Machine Learning Repository, University of California, Irvine, School of Information and Computer Sciences, 2021, URL http://archive.ics.uci.edu/ml.
[46] V. Vovk, Kernel ridge regression, in: B. Schölkopf, Z. Luo, V. Vovk (Eds.), Empirical Inference: Festschrift in Honor of Vladimir N. Vapnik, Springer Berlin Heidelberg, Berlin, Heidelberg, 2013, pp. 105–116.
[47] P.N. Suganthan, On non-iterative learning algorithms with closed-form solution, Appl. Soft Comput. 70 (2018) 1078–1082.
[48] L.L.C. Kasun, H. Zhou, G.-B. Huang, C.M. Vong, Representational learning with ELMs for big data, 2013.
[49] J. Demšar, Statistical comparisons of classifiers over multiple data sets, J. Mach. Learn. Res. 7 (2006) 1–30.
[50] J.D. Gibbons, S. Chakraborti, Nonparametric Statistical Inference, CRC Press, 2014.