On the Use of Modular Indistinguishability Operators in RBFNN-like Models*

Alberto Ortiz[1,2,3][0000−0002−8253−7455], Óscar Valero[1,2][0000−0003−4710−1338], and Juan José Miñana[4][0000−0001−9835−0700]

[1] Dep. Mathem. and Computer Science, Univ. of the Balearic Islands (UIB), Spain
[2] IDISBA (Health Research Institute of the Balearic Islands), Spain
[3] IAIB (Artificial Intelligence Research Institute of the UIB), Spain
[4] Dep. Applied Mathematics, Polytechnic University of Valencia, Spain
{alberto.ortiz,oscar.valero}@uib.es, juamiapr@mat.upv.es

Abstract. Radial Basis Function Neural Networks (RBFNN) have become popular machine learning models with a simple structure but at the same time strong non-linear function approximation and effective modeling capabilities. In this work, we explore the use of *Modular Indistinguishability Operators* (MIO) in RBFNN-like structures to replace the RBFs that populate the hidden layer, to give rise to *MIO-based Neural Networks* (MIO-NN). In this respect, we introduce a new distance function and prove that it is a modular metric, to next use it to derive two MIOs to be evaluated as the key component of MIO-NNs. As an additional contribution, we describe *Self-Defining MIO-NN* (SD-MIO-NN) as an approach capable of configuring MIO-NNs in a parameterless way. SD-MIO-NN comprises a first step that defines the size of the hidden layer, a second step that determines the parameters of the hidden neurons and a last step that calculates the weights of the hidden-to-output layer connections. The experimental results show the effectiveness of the proposed MIOs for multi-class classification, and by extension of SD-MIO-NN, which in turn compares well with other similar solutions.

Keywords: Multi-class Classification · RBF Neural Networks (RBFNN) · Modular Indistinguishability Operators (MIO).

1 Introduction

As part of the continuous development of problem solving methodologies based on artificial intelligence, Radial Basis Function Neural Networks (RBFNN) have become popular machine learning models with a simple structure but at the same

* This work is partially supported by EU-H2020 grant BUGWRIGHT2 (GA 871260) and by grant PID2022-139248NB-I00 (funded by MCIN/AEI/10.13039/501100011033 and by "ERDF A way of making Europe"). This publication reflects only the authors views and the European Union is not liable for any use that may be made of the information contained therein.

time strong non-linear function approximation and effective modeling capabilities [14]. This has permitted to address both regression and classification problems, and in fields such as non-linear modeling [22, 29, 11] and forecasting [13, 30, 17], non-linear control [9, 23, 2] and pattern recognition [4, 25, 27, 1], to name but a few. Besides, this is achieved with a considerably low computational cost, suitable for energy-limited scenarios, e.g. edge computing devices.

RBFNNs are single-hidden layer neural networks where *radial-basis functions* (RBF) play the key role. An RBF is a real-valued function $\rho(z)$ whose value depends only on the distance $d(x, \mu)$ between input samples x and a specific point μ that is associated with each hidden unit and it is known as its *center*, so that $z = d(x, \mu)$. Apart from other RBFs [14], the RBF most usually adopted is the Gaussian function $\rho(r) = \exp\left(-\frac{r^2}{2\sigma^2}\right)$, where $r^2 = \|x - \mu\|_2^2$ and σ is known as the *width* of the unit. An RBF, as it is defined, can also be regarded as a *similarity function* [24], i.e. a function $\rho(x, \mu)$ that measures how similar are x and μ so that $\rho \to \widehat{s}$ as $r \to 0$ and $\rho \to 0$ as $r \to \infty$ (with $\widehat{s}$ typically 1).

In this work, we break the typical dependence of ρ on $r = \|x - \mu\|_2$ and consider the use of more generic similarity functions, i.e. $\rho \to \widehat{s}$ as $d(x, \mu) \to \widehat{d}$ (typically $\widehat{d} = 0$) and $\rho \to \bar{s} < \widehat{s}$ (typically, $\bar{s} = 0$) as $d(x, \mu) \to \bar{d} > \widehat{d}$ for a given *dissimilarity function* d [24]. Actually, inspired by the distance function proposed in [20], we define a *modular metric* m [3] from which we derive several *Modular Indistinguishability Operators* (MIO) I_T [18], which are in turn integrated in the hidden neurons of an RBFNN-like network, hence introducing *MIO-based Neural Networks* (MIO-NN) as an alternative to the RBFNN concept. In this paper, we evaluate the performance resulting from these MIOs within the framework of MIO-NNs in multi-class classification tasks.

In what follows, we enumerate the main contributions of this work:

- Within the context of multi-class classification tasks, we explore the use of Modular Indistinguishability Operators (MIO) in RBFNN-like structures to replace the RBFs that populate the hidden layer, to give rise to *MIO-based Neural Networks* (MIO-NNs).
- We revise the distance function proposed in [20] and introduce a new metric m, to next use it to derive two MIOs to be evaluated as the key component of MIO-NNs. In the appendix, we prove that m is a modular metric.
- We describe SD-MIO-NN as an approach capable of defining MIO-NNs in a parameterless way.
- The experimental setup for evaluating SD-MIO-NN takes into consideration a total of 17 datasets comprising samples of a quite diverse nature/source. The experimental results obtained show competitive performance for MIO-NNs, and by extension for SD-MIO-NN, which also compares well with other single-hidden layer algorithms.

The rest of the paper is organized as follows: Section 2 reviews the RBFNN concept and links it to MIOs; Section 3 revises a known distance function, introduces a new modular metric taking inspiration from such a distance, and finally derives a number of MIOs from it; Section 4 describes SD-MIO-NN in detail;

Section 5 reports on the effectiveness of the MIOs defined in Section 3 as well as on the performance attained by SD-MIO-NN on a number of publicly available datasets; finally, Section 6 concludes the paper summarizing the main findings and outlining future work.

2 Preliminaries

A *Radial Basis Function Neural Network* (RBFNN) is a special class of artificial neural network [19], whose main differences with *Feed-Forward Neural Networks* (FFNN) are the fact that an RBFNN typically comprises, apart from the input and output layers, a single hidden layer, and the fact that the hidden neurons implement an instance of a radial basis function $\rho(z)$ instead of the usual dot product $w^T x$ between connection weights w and the neuron input x. The value of an RBF depends on the distance from x to the *center vector* or *prototype* μ_j associated to the neuron, i.e. $\rho(z) = \rho(d(x, \mu_j))$, and hence they become radially symmetric about μ_j. Common elections are the Euclidean norm for the distance $d(x, \mu_j)$ and a Gaussian as the radial basis function:

$$\rho\big(d(x,\mu_j)\big) = \rho\big(\,\|x-\mu_j\|_2\,\big) = \exp\left[-\frac{\|x-\mu_j\|_2^2}{2\sigma_j^2}\right] = \exp\left[-\frac{r_{x,j}^2}{2\sigma_j^2}\right] \tag{1}$$

where σ_j is the *width* of the hidden unit j. In the context of a classification problem involving C classes, output Φ_k (for class $k = 1, \ldots, C$) of an RBFNN with n_h hidden neurons is then given by:

$$\Phi_k(x) = \sum_{j=1}^{n_h} w_{k,j}\ \rho\big(d(x,\mu_j)\big) + b_k \tag{2}$$

with b_k as the bias of the output neuron k.

Other RBFs considered in the literature are e.g. the *inverse multiquadric* $\rho(r) = 1/\sqrt{1 + r^2/c^2}$, for $c > 0$, and the *logistic function* $\rho(r) = 2/(1 + \exp[r^2/\sigma^2])$ [21, 10]. These functions, together with the Gaussian of Eq. (1), behave as *similarity functions* and produce values in the interval $[0, 1]$, with $\rho \to \widehat{s} = 1$ as $r \to \widehat{d} = 0$ and $\rho \to \bar{s} = 0$ as $r \to \bar{d} = \infty$.

In this paper, we consider several *modular indistinguishability operators* that behave as similarity functions in the sense outlined above, and also depend on a dissimilarity function. Because of the dissimilarity function adopted, the receptive fields of the hidden neurons are not hyper-spherical, as corresponds to the Gaussian RBF, but they exhibit a different shape and properties.

3 Modular Indistinguishability Operators (MIO)

In this section, we introduce a new modular metric which is used to build two *Modular Indistinguishability Operators* (MIO) in the sense of [18] by means of the technique described by Miñana and Valero in [18, Theorem 5].

3.1 A modular metric on $[0, \infty[^n$

The mapping $d_0 : [0,1]^n \times [0,1]^n \to [0,1]$ given by $d_0(\vec{0}, \vec{0}) = 0$ and

$$d_0(p,q) = \frac{\sum_{i=1}^{n} |p_i - q_i|}{\sum_{i=1}^{n} \max\{p_i, q_i\}} \text{ elsewhere}, \tag{3}$$

for each $p, q \in [0,1]^n$ with $p = (p_1, \ldots, p_n)$ and $q = (q_1, \ldots, q_n)$, was already introduced in the literature and proved to be a *metric* in [20]. Nevertheless, it is not hard to check that d_0 is also a metric on D^n, where $D = [0, \infty[$ and $n \in \mathbb{N}$. Inspired by d_0, we define the mapping $m :]0, \infty[\times D^n \times D^n \to [0,1]$ given by

$$m(t,p,q) = \frac{\sum_{i=1}^{n} |p_i - q_i|}{t + \sum_{i=1}^{n} \max\{p_i, q_i\}} \text{ for each } t \in]0, \infty[\text{ and } p, q \in D^n\,. \tag{4}$$

In the Appendix, we demonstrate that m is a modular metric on D^n.

3.2 Modular indistinguishability operators built from m

In this section, we present two modular indistinguishability operators I_T that can be induced from the modular metric m, being T a continuous Archimedean t-norm with f as an additive generator (for a detailed treatment on t-norms theory we refer the reader to [15]). Both MIOs have been generated by means of the method based on the use of the pseudo-inverse of the additive generator f introduced in [18, Theorem 5], which can be stated as follows: *Let (X, m) be a modular metric space and let T be a continuous Archimedean t-norm with additive generator f. Then, $I_T = (f^{(-1)} \circ m)$ is a T-modular indistinguishability operator on X.*

Modular indistinguishability operators on D^n for the product t-norm Let T_P be the product t-norm. It is well-known that an additive generator of T_P is given by $f_P(x) = -(\log x)$ for all $x \in [0,1]$, and that the pseudo-inverse of this additive generator is given by $f_P^{(-1)}(y) = \mathrm{e}^{-y}$ for all $x \in [0, \infty]$. Then, for each $t \in]0, \infty[$ and $p, q \in D^n$ we have:

$$I_P(t,p,q) = \left(f_P^{(-1)} \circ m\right)(t,p,q) = \mathrm{e}^{-m(t,p,q)} \tag{5}$$

Modular indistinguishability operators on D^n for a Hamacher t-norm ($\lambda = 0$) We first recall that the family $\left(T_H^\lambda\right)_{\lambda \in [0,\infty]}$ of Hamacher t-norms is given by

$$T_H^\lambda(x,y) = \begin{cases} T_D(x,y), & \text{if } \lambda = \infty \\ 0, & \text{if } \lambda = x = y = 0 \\ \frac{xy}{\lambda + (1-\lambda)(x+y-xy)} & \text{otherwise} \end{cases} \tag{6}$$

For our construction we only consider the Hamacher t-norm with $\lambda = 0$, i.e., $T_H(x,y) = \frac{xy}{x+y-xy}$ (note that $T_P \leq T_H$). An additive generator of T_H is

given by $f_H(x) = \frac{1-x}{x}$ for all $x \in [0,1]$, and the pseudo-inverse of that additive generator is given by $f_H^{(-1)}(y) = \frac{1}{1+y}$ for all $x \in [0,\infty]$. Then, for each $t \in]0,\infty[$ and $p, q \in D^n$ we have

$$I_H(t,p,q) = \left(f_H^{(-1)} \circ m\right)(t,p,q) = \frac{1}{1+m(t,p,q)} \tag{7}$$

4 Overview of SD-MIO-NN

To build MIO-NN models, we define the SD-MIO-NN (*Self-Defining MIO-NN*) algorithm, which comprises three steps: (1) the training data is clustered by means of a *Self-Organizing Map* (SOM) to locate areas in feature space populated by samples; (2) the hidden layer of the MIO-NN is set up with as many neurons as clusters are defined in the SOM after training, using the cluster representatives as neuron centers μ_j, while the neuron widths σ_j are set on the basis of the inter-sample distances between centers, and ρ in Eq. (2) is defined as a MIO I_T; (3) the hidden layer-to-output layer weights are found by means of regularized least squares. Additional details can be found in the following sections.

SOM building and training A SOM is a special class of artificial neural network that comprises a single computational layer of neurons arranged in a lattice, i.e. each hidden neuron n_{ij} is assigned a cell (i,j) in a grid. Using a combination of competitive and cooperative training, SOM neurons learn a weight vector ω_{ij} with the same dimensionality as the input samples $x \in \mathbb{R}^n$ which represent populated areas of the input space. After training, the network is expected to get organized so that neighbouring samples in the input space map to the same or neighbouring cells in the lattice. Accordingly, samples are arranged into clusters/classes through the grid. (For more details, see [16] among many others.)

In SD-MIO-NN, the SOM is defined as a 2D rectangular grid with a number of neurons n_{som} such that $n_{\text{som}} = 5\sqrt{N}$ for N training samples. The size $n_1 \times n_2$ of the SOM grid is defined by the ratio of the two largest eigenvalues λ_1, λ_2 of the covariance matrix of the training set (i.e. $n_1 \times n_2 \approx n_{\text{som}}, \lambda_1/\lambda_2 = n_1/n_2$), while the initial weights $\omega_{ij}(0)$ for each neuron are determined using PCA. This defines a rectangular grid whose directions and extent match the two main directions and range of the data, so as to initialize the grid weights ω_{ij} with points of the feature space regularly spaced throughout the input data subspace. Due to the way how the SOM is defined and trained, the initialization step of SD-MIO-NN does not require setting any critical parameter, in particular the number of clusters, unlike, for example, K-means and many others, or setting up any other parameter e.g. related to the density of samples (in DBSCAN and others). Moreover, the SOM makes use of the same distance function adopted in the hidden layer neurons, to be coherent in the measurement of dissimilarity from end to end.

MIO-NN definition Once the SOM has been built, each training sample x is associated to the SOM neuron whose weight ω_{ij} is closest to x, i.e. its BMU (*best maching unit*), what leads to *non-empty* and *empty cells* depending on whether any training sample has got associated to the neuron or not, i.e. *contain* samples (it can happen that some cells, though initialized with a reasonable weight $\omega_{ij}(0)$, do not get associated to any sample and thus get *empty*). After this association process, each *non-empty* cell of the SOM gives rise to a neuron n_k in the hidden layer of the MIO-NN, whose center μ_k is the final weight vector $\omega_{ij}(t_{\max})$ attained by that SOM cell.

Regarding the neuron widths σ_k of standard RBFNNs, they do not appear in the MIOs of Eqs. (5) and (7) as explicitly as in the case of a Gaussian RBF. However, parameter t of the modular metric m, Eq. (4), can be related to the size of the receptive field of the hidden unit when m is written as follows:

$$m(t,p,q) = \frac{\frac{\sum_{i=1}^{n} |p_i - q_i|}{t}}{1 + \frac{\sum_{i=1}^{n} \max\{p_i, q_i\}}{t}} \,. \tag{8}$$

In what follows, we will keep using σ to denote the width of the hidden unit (in order not to change the usual nomenclature), although it must be understood that $\sigma_k = t_k$ for neuron k when the modular metric m is used. Further, inspired by [19], widths are automatically defined as $\sigma_j = \alpha_w \, d_{\text{avg}}, \forall j$, where d_{avg} is the average of the distances between every center μ_j and its nearest center $\mu_{j\prime}$, and $\alpha_w \in [1, 2]$ is a parameter.

Output layer training SD-MIO-NN adopts an uncoupled approach to train the model, which means that the hidden-to-output layer connection weights $w_{k,j}$ in Eq. (2) are determined separately once neuron centers and widths have been set. Weights $w_{k,j}$ are found through the following regularized least-squares formulation to counteract overfitting:

$$\min \; \frac{1}{2} \sum_{i=1}^{N} \sum_{k=1}^{C} \left(y_{i,k} - \left(\sum_{j=1}^{n_h} w_{k,j} \; \rho\big(d(x, \mu_j)\big) + b_k \right) \right)^2 + \lambda \frac{1}{2} \sum_{k=1}^{C} \sum_{j=1}^{n_h} w_{k,j}^2 \tag{9}$$

where n_h is the number of neurons of the hidden layer, C is the number of classes and $y_{i,k}$ is the one-hot encoding of class labels and λ is the regularization factor.

5 Experimental Results

In the following, we evaluate the performance of SD-MIO-NN within the context of multi-class supervised classification tasks using standard metrics and publicly available well-known datasets. In more detail, classification performance is reported in terms of accuracy (A) and the F_1 score, which is the harmonic mean of the precision (P) and recall (R) metrics:

$$\mathrm{A} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{N}} \,, \; \mathrm{P} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} \,, \; \mathrm{R} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \,, \; \mathrm{F}_1 = \frac{2\mathrm{PR}}{\mathrm{P} + \mathrm{R}}$$

Table 1. Main features of the datasets considered. (IR stands for *imbalance ratio*, expressed by the quotient *no. samples majority class / no. samples minority class* [28].)

dataset	no. features	no. classes	no. samples	IR
1. Iris	4	3	150	1.00
2. Penguins	6	3	333	2.15
3. Wine	13	3	178	1.50
4. Wisconsin breast cancer (original, WBC-org)	9	2	683	1.86
5. BUPA liver disorders	6	2	345	1.38
6. PIMA Indians Diabetes	8	2	768	1.87
7. Lower Back Pain Symptoms (LBPS)	11	2	310	2.10
8. Phoneme	5	2	5404	2.41
9. Segment	19	7	2310	1.00
10. Wisconsin breast cancer (WBC)	30	2	569	1.68
11. Satimage	36	6	6430	2.45
12. Digits	64	10	1797	1.05
13. DNA	180	2	3186	2.16
14. Sonar	60	2	207	1.16
15. Heart	13	5	297	12.31
16. Glass	9	6	214	8.44
17. Yeast	8	10	1484	92.6

where TP, TN, FP and FN are, respectively, the true positives, true negatives, false positives and false negatives, and N is the total number of samples. All the performance values reported in this section correspond to the average of the metric values after stratified 5-fold cross validation, using *weighted-averaging* for the multi-class datasets (that is to say, the weights match the support for each class, i.e. the number of true instances). In all cases, we use $\alpha_w = 1.2$ (Section 4) and $\lambda = 0.01$ (Eq. (9)).

The evaluation of SD-MIO-NN considers a benchmark comprising several sorts of datasets, whose main features are summarized in Table 1, grouped in a first *basic set* (datasets 1-3), followed by a second set of *two-class (moderately) imbalanced* datasets [26] (datasets 4-8), datasets comprising *high-dimensional* samples (datasets 9-14) and a final subset with *highly imbalanced* classes (datasets 15-17). All datasets can be found in public repositories, e.g. UCI-ML, Kaggle or DataHub. All of them have been normalized (using *max-min* normalization) before training and building the MIO-NN through SD-MIO-NN.

Table 2 collects relevant data on the performance of SD-MIO-NN for all the datasets and considering the two MIOs I_H and I_P introduced in Section 3: size of the SOM involved in the first step, number of hidden neurons resulting from the second step, and classification accuracy and F_1 values, given as minimum and maximum values attained, average value and standard deviation, all calculated over the results of the 5 folds of the cross-validation process. We also compare the performance achieved with a Gaussian RBFNN that is fed by the SOM-

based initialization step of SD-MIO-NN. As can be observed, the resulting MIO-NN models outperform the model making use of the Gaussian RBF and the Euclidean distance in all datasets. As a final comment, notice that the differences that can be observed in the size of the hidden layer (column NH in Table 2) come from the fact that SD-MIO-NN applies the same distance function when building both the SOM and the hidden layer: i.e. m as defined in Eq. (4) when using the MIOs, and the Euclidean distance when using the Gaussian RBF.

To finish, Table 3 compares SD-MIO-NN with other approaches in terms of classification accuracy and number of hidden neurons: GAP-RBF [12], FGAP-RBF [31], AANN [8], CP-NN [7], BeeRBF [5] and GA-IPSO-CSRBF [28]. We include them in the comparison because they all deal with single-hidden layer networks focusing on classification (it must be noticed that this is not the typical situation of RBFNNs since, in general, these models focus on regression, e.g. [6]). Furthermore, the above-mentioned methods have been published as self-adjusting schemes, i.e. they apply iterative processes of refinement to adjust the structure of the network to the task at hand; this is the reason why some of them finish with a low number of neurons in general, e.g. AANN. The accuracy values that we report come directly from the original publications, which do not show F_1 values and hence we cannot include them in Table 3.

From the table, one can observe that SD-MIO-NN outperforms other solutions, i.e. the highly imbalanced datasets *heart* and *yeast*, the high-dimensional dataset *sonar*, and the datasets *wine* and *wbc-org*, or achieves almost the same performance level, i.e. datasets *iris*, *bupa*, *pima* and *wbc*. As for the size of the hidden layer, SD-MIO-NN does not optimize it as already discussed, contrary to the rest of algorithms included in the comparison; nonetheless, SD-MIO-NN, even given the aforementioned, gives rise to contained layer sizes and even reduces considerably the number of neurons for some datasets, i.e. *phoneme*, *satimage* and *dna*, achieving similar performance than models produced by other methods.

6 Conclusions and Future Work

In this work, we have explored the use of *Modular Indistinguishability Operators* (MIO) in RBFNN-like architectures to replace the Gaussian RBFs that normally populate the hidden layer in this kind of structures, to give rise to *MIO-based Neural Networks* (MIO-NN). In this respect, we have introduced a modular metric m on $[0, \infty[^n$ to next use it to define MIOs that can be integrated in MIO-NNs. As an additional contribution, we have described SD-MIO-NN as an approach capable of defining MIO-NNs in a parameterless way.

SD-MIO-NN has been evaluated in the context of multi-class classification tasks by means of a total of up to 17 datasets featuring different number of classes, imbalance ratios and number of dimensions. SD-MIO-NN in combination with the different MIOs has been shown to be able to attain a high level of performance, with slight differences among the MIOs considered. SD-MIO-NN has also shown competitive performance against other algorithms for single-hidden layer neural networks.

Table 2. Performance achieved by MIO-NNs populated with I_P and I_H neurons (Eqs. 5 and 7) against a Gaussian RBFNN, for all datasets of the benchmark. (SOM: size of the SOM, $n_1 \times n_2$; NH: nr. of neurons of the hidden layer, n_h; AMM: A max/min; AAS: A avg $\pm$ std; FMM: F_1 max/min; FAS: F_1 avg $\pm$ std)

			MIO-NN + I_P				MIO-NN + I_H			
dataset	**SOM**	**NH**	**AMM**	**AAS**	**FMM**	**FAS**	**AMM**	**AAS**	**FMM**	**FAS**
iris	3 × 21	51	0.97/0.90	0.95 ± 0.03	0.97/0.90	0.95 ± 0.03	0.97/0.90	0.95 ± 0.03	0.97/0.90	0.95 ± 0.03
penguins	6 × 16	79	0.98/0.95	0.98 ± 0.01	0.98/0.96	0.98 ± 0.01	0.98/0.95	0.98 ± 0.01	0.98/0.96	0.98 ± 0.01
wine	6 × 12	55	1.00/0.91	0.97 ± 0.03	1.00/0.91	0.97 ± 0.03	1.00/0.94	0.98 ± 0.02	1.00/0.94	0.98 ± 0.02
wbc-org	4 × 33	98	0.99/0.96	0.97 ± 0.01	0.99/0.96	0.97 ± 0.01	0.99/0.96	0.97 ± 0.01	0.99/0.96	0.97 ± 0.01
bupa	7 × 14	84	0.72/0.59	0.66 ± 0.04	0.72/0.58	0.65 ± 0.05	0.72/0.59	0.65 ± 0.04	0.71/0.58	0.64 ± 0.04
pima	10 × 14	125	0.80/0.71	0.75 ± 0.03	0.79/0.70	0.75 ± 0.03	0.80/0.73	0.76 ± 0.02	0.79/0.72	0.75 ± 0.02
lbps	10 × 9	85	0.79/0.67	0.72 ± 0.04	0.78/0.65	0.70 ± 0.05	0.79/0.67	0.72 ± 0.04	0.78/0.65	0.71 ± 0.05
phoneme	11 × 15	121	0.87/0.77	0.83 ± 0.03	0.86/0.77	0.83 ± 0.03	0.87/0.79	0.83 ± 0.03	0.86/0.78	0.83 ± 0.03
segment	5 × 22	82	0.95/0.89	0.92 ± 0.02	0.94/0.89	0.91 ± 0.02	0.95/0.89	0.91 ± 0.02	0.94/0.89	0.91 ± 0.02
wbc	7 × 18	117	0.98/0.94	0.96 ± 0.02	0.98/0.94	0.96 ± 0.02	0.98/0.93	0.96 ± 0.02	0.98/0.93	0.96 ± 0.02
satimage	11 × 17	151	0.89/0.86	0.88 ± 0.01	0.89/0.86	0.87 ± 0.01	0.90/0.86	0.87 ± 0.02	0.90/0.85	0.87 ± 0.02
digits	14 × 16	188	0.90/0.85	0.87 ± 0.01	0.90/0.85	0.87 ± 0.02	0.90/0.85	0.87 ± 0.02	0.90/0.85	0.87 ± 0.02
dna	11 × 12	86	0.93/0.86	0.89 ± 0.03	0.93/0.86	0.89 ± 0.03	0.93/0.85	0.88 ± 0.03	0.93/0.85	0.88 ± 0.03
sonar	8 × 9	53	0.90/0.80	0.83 ± 0.04	0.90/0.80	0.83 ± 0.04	0.88/0.75	0.80 ± 0.04	0.88/0.74	0.80 ± 0.04
heart	8 × 11	37	0.66/0.55	0.59 ± 0.04	0.60/0.47	0.51 ± 0.05	0.61/0.55	0.58 ± 0.02	0.53/0.47	0.50 ± 0.02
glass	6 × 13	55	0.79/0.64	0.72 ± 0.05	0.74/0.61	0.68 ± 0.05	0.79/0.64	0.73 ± 0.06	0.75/0.62	0.70 ± 0.05
yeast	10 × 20	195	0.61/0.56	0.59 ± 0.02	0.58/0.55	0.57 ± 0.01	0.61/0.56	0.58 ± 0.02	0.58/0.54	0.56 ± 0.01

			Gaussian RBFNN			
dataset	**SOM**	**NH**	**AMM**	**AAS**	**FMM**	**FAS**
iris	3 × 21	50	0.93/0.86	0.90 ± 0.03	0.93/0.86	0.90 ± 0.03
penguins	6 × 16	78	0.92/0.82	0.88 ± 0.04	0.92/0.81	0.88 ± 0.04
wine	6 × 12	58	0.62/0.40	0.53 ± 0.08	0.61/0.34	0.49 ± 0.11
wbc-org	4 × 33	109	0.96/0.90	0.93 ± 0.02	0.96/0.90	0.93 ± 0.02
bupa	7 × 14	89	0.63/0.51	0.58 ± 0.04	0.63/0.50	0.57 ± 0.04
pima	10 × 14	126	0.65/0.59	0.63 ± 0.02	0.60/0.53	0.56 ± 0.03
lbps	10 × 9	82	0.79/0.59	0.66 ± 0.07	0.75/0.50	0.61 ± 0.09
phoneme	11 × 15	120	0.77/0.72	0.74 ± 0.02	0.72/0.65	0.69 ± 0.03
segment	5 × 22	77	0.65/0.54	0.60 ± 0.04	0.67/0.53	0.61 ± 0.05
wbc	7 × 18	119	0.72/0.64	0.68 ± 0.03	0.68/0.55	0.63 ± 0.04
satimage	11 × 17	156	0.37/0.30	0.34 ± 0.03	0.32/0.21	0.27 ± 0.04
digits	14 × 16	188	0.10/0.10	0.10 ± 0.00	0.02/0.02	0.02 ± 0.00
dna	11 × 12	125	0.52/0.52	0.52 ± 0.00	0.36/0.36	0.36 ± 0.00
sonar	8 × 9	60	0.60/0.54	0.56 ± 0.02	0.49/0.37	0.42 ± 0.05
heart	8 × 11	42	0.57/0.47	0.52 ± 0.03	0.46/0.36	0.40 ± 0.03
glass	6 × 13	48	0.67/0.54	0.59 ± 0.05	0.66/0.53	0.57 ± 0.05
yeast	10 × 20	164	0.41/0.31	0.35 ± 0.03	0.39/0.27	0.32 ± 0.04

Table 3. Accuracy (1st row) and number of hidden neurons (2nd row) of SD-MIO-NN against other single hidden layer approaches. (The best result is indicated in red.)

dataset	*GAP-RBF* [12]	*FGAP-RBF* [31]	*AANN* [8]	*CP-NN* [7]	*BeeRBF* [5]	*GA-IPSO-...* [28]	*SD-MIO-NN* I_P	*SD-MIO-NN* I_H
iris	–	–	–	0.95	0.96	–	0.95	0.95
	–	–	–	21.0	11.6	–	51.0	51.0
wine	–	–	–	–	0.97	–	0.97	0.98
	–	–	–	–	18.7	–	55.0	55.0
wbc-org	–	–	–	–	–	0.96	0.97	0.97
	–	–	–	–	–	–	98.0	98.0
bupa	–	–	–	–	–	0.68	0.66	0.65
	–	–	–	–	–	–	84.0	84.0
pima	–	–	–	–	0.77	0.74	0.75	0.76
	–	–	–	–	35.8	–	125.0	125.0
lbps	–	–	–	–	–	0.81	0.72	0.72
	–	–	–	–	–	–	85.0	85.0
phoneme	0.77	0.87	–	–	–	–	0.83	0.83
	150.4	996.5	–	–	–	–	121.0	121.0
segment	0.89	0.96	–	–	–	–	0.92	0.91
	42.7	423.9	–	–	–	–	82.0	82.0
wbc	–	–	0.67	0.66	0.97	0.96	0.96	0.96
	–	–	9.0	22.0	71.2	–	117.0	117.0
satimage	–	0.92	–	–	–	–	0.88	0.87
	–	1455.0	–	–	–	–	151.0	151.0
dna	–	0.93	–	–	–	–	0.89	0.88
	–	638.4	–	–	–	–	86.0	86.0
sonar	–	–	–	–	0.59	–	0.83	0.80
	–	–	–	–	40.0	–	53.0	53.0
heart	–	–	–	–	0.53	–	0.59	0.58
	–	–	–	–	11.4	–	37.0	37.0
glass	–	–	–	–	0.73	–	0.72	0.73
	–	–	–	–	18.6	–	55.0	55.0
yeast	–	–	0.55	0.52	–	–	0.59	0.58
	–	–	9.0	43.0	–	–	195.0	195.0

'–' means the value is not reported in the corresp. publication.

As for future work, the adaptation and evaluation of SD-MIO-NN for regression tasks is a topic we are working on at the moment. On the other side, it is under study the adoption of other modular metrics and MIOs for enhanced net properties and performance, as well as the development of other methodologies for linking the parameter t of the modular metric with the classification/regression task to solve.

Appendix: Proof that m is a modular metric

According to V.V. Chytiakov (see [3]), a function $w :]0, \infty[\times X \times X \to [0, \infty]$ is a *modular metric* on a non-empty set X if for each $x, y, z \in X$ and each $t, s \in]0, \infty[$ the following axioms are fulfilled:

(MM1) $w(t, x, y) = 0$ for all $t \in]0, \infty[$ if and only if $x = y$;

(MM2) $w(t,x,y) = w(t,y,x)$;
(MM3) $w(t+s,x,z) \leq w(t,x,y) + w(s,y,z)$.

In such a case, we say that (X,w) is a modular metric space.

Let $m :]0,\infty[\times D^n \times D^n \to [0,1]$ be the mapping given in (4). To prove that m is a modular metric, we just need to prove that **(MM3)** is satisfied, since showing that **(MM1)** and **(MM2)** are fulfilled by m is immediate.

Now, let $t, s \in]0,\infty[$ and $p,q,r \in D^n$, where $p = (p_1,\ldots,p_n)$, $q = (q_1,\ldots,q_n)$ and $r = (r_1,\ldots,r_n)$. Fix $\lambda = \max\{t,s\}$. Next we prove that

$$m(\lambda,p,q) \leq m(\lambda,p,r) + m(\lambda,r,q).$$

Observe that the function $m_{p,q} :]0,\infty[\to [0,1]$ is non-increasing, where $m_{p,q}(t) = m(t,p,q)$ for all $t \in]0,\infty[$. Thus the previous inequality yields the next one

$$m(t+s,p,q) \leq m(\lambda,p,q) \leq m(\lambda,p,r) + m(\lambda,r,q) \leq m(t,p,r) + m(s,r,q),$$

which shows that **(MM3)** is fulfilled.

We distinguish two possible cases:

Case 1: $r_i \leq \max\{p_i,q_i\}$ for all $i \in \{1,\ldots,n\}$. Then, $\max\{p_i,r_i\} \leq \max\{p_i,q_i\}$ and $\max\{r_i,q_i\} \leq \max\{p_i,q_i\}$ for all $i \in \{1,\ldots,n\}$. Therefore,

$$m(\lambda,p,r) + m(\lambda,r,q) = \frac{\sum_{i=1}^{n} |p_i - r_i|}{\lambda + \sum_{i=1}^{n} \max\{p_i,r_i\}} + \frac{\sum_{i=1}^{n} |r_i - q_i|}{\lambda + \sum_{i=1}^{n} \max\{r_i,q_i\}} \geq$$

$$\geq \frac{\sum_{i=1}^{n} |p_i - r_i|}{\lambda + \sum_{i=1}^{n} \max\{p_i,q_i\}} + \frac{\sum_{i=1}^{n} |r_i - q_i|}{\lambda + \sum_{i=1}^{n} \max\{p_i,q_i\}} =$$

$$= \frac{\sum_{i=1}^{n} (|p_i - r_i| + |r_i - q_i|)}{\lambda + \sum_{i=1}^{n} \max\{p_i,q_i\}} \geq \frac{\sum_{i=1}^{n} |p_i - q_i|}{\lambda + \sum_{i=1}^{n} \max\{p_i,q_i\}} = m(\lambda,p,q).$$

Case 2: There exists (at least one) $i_0 \in \{1,\ldots,n\}$ such that $r_{i_0} > \max\{p_{i_0},q_{i_0}\}$.
In the subsequent argument, the next lemma, whose easy proof we omit, plays a key role.

Lemma 1 *Let $\alpha,\beta \geq 0$ and $\gamma,\lambda \geq 0$, with $\gamma + \lambda > 0$ and $\gamma \geq \alpha$. We define a function $f_{\alpha,\beta,\gamma,\lambda} : [\beta,\infty[\to D$ given by*

$$f_{\alpha,\beta,\gamma,\lambda}(x) = \frac{x - \beta + \alpha}{\lambda + x + \gamma},$$

for each $x \in [\beta,\infty[$. Then f is non-decreasing.

Now let $I \subset \{1, \ldots, n\}$ be the set of indices such that $r_i > \max\{p_i, q_i\}$ for all $i \in I$ and $J = \{1, \ldots, n\} \setminus I$. Note that we have $r_j \leq \max\{p_j, q_j\}$ for each $j \in J$.
Let $r' = (r'_1, \ldots, r'_n)$, where

$$r'_i = \begin{cases} \max\{p_i, q_i\}, & \text{if} \quad i \in I \\ r_i, & \text{if} \quad i \in J \end{cases}.$$

Note that for each $i \in I$ we have $\max\{p_i, r'_i\} = r'_i = \max\{r'_i, q_i\}$ and so $r_i > r'_i$.
Clearly p, q and r' satisfy the condition of Case 1, i.e., $r'_i \leq \max\{p_i, q_i\}$ for all $i \in \{1, \ldots, n\}$. Hence

$$m(\lambda, p, q) \leq m(\lambda, p, r') + m(\lambda, r', q).$$

Next, we show that $m(\lambda, p, r') \leq m(\lambda, p, r)$.

$$m(\lambda, p, r) = \frac{\sum\limits_{i=1}^{n} |p_i - r_i|}{\lambda + \sum\limits_{i=1}^{n} \max\{p_i, r_i\}} = \frac{\sum\limits_{i \in I} |p_i - r_i| + \sum\limits_{j \in J} |p_j - r_j|}{\lambda + \sum\limits_{i \in I} \max\{p_i, r_i\} + \sum\limits_{j \in J} \max\{p_j, r_j\}} =$$

$$= \frac{\sum\limits_{i \in I} r_i - \sum\limits_{i \in I} p_i + \sum\limits_{j \in J} |p_j - r_j|}{\lambda + \sum\limits_{i \in I} r_i + \sum\limits_{j \in J} \max\{p_j, r_j\}}.$$

In the same manner, we can verify that

$$m(\lambda, p, r') = \frac{\sum\limits_{i \in I} r'_i - \sum\limits_{i \in I} p_i + \sum\limits_{j \in J} |p_j - r'_j|}{\lambda + \sum\limits_{i \in I} r'_i + \sum\limits_{j \in J} \max\{p_j, r'_j\}} =$$

$$\frac{\sum\limits_{i \in I} r'_i - \sum\limits_{i \in I} p_i + \sum\limits_{j \in J} |p_j - r_j|}{\lambda + \sum\limits_{i \in I} r'_i + \sum\limits_{j \in J} \max\{p_j, r_j\}}.$$

Let $\beta = \sum\limits_{i \in I} p_i$, $\alpha = \sum\limits_{j \in J} |p_j - r_j|$ and $\gamma = \sum\limits_{j \in J} \max\{p_j, r_j\}$. Then we have that $\alpha, \beta \geq 0$, $\gamma + \lambda > 0$ and $\gamma \geq \alpha$.
Since $r_i > r'_i \geq p_i$ for each $i \in I$, we obtain that $\sum\limits_{i \in I} r_i > \sum\limits_{i \in I} r'_i \geq \sum\limits_{i \in I} p_i$. Thus $\sum\limits_{i \in I} r_i, \sum\limits_{i \in I} r'_i \in [\sum\limits_{i \in I} p_i, \infty[$. Therefore, by Lemma 1, we deduce that

$$m(\lambda, p, r) = f_{\alpha,\beta,\gamma,\lambda}\left(\sum_{i \in I} r_i\right) \geq f_{\alpha,\beta,\gamma,\lambda}\left(\sum_{i \in I} r'_i\right) = m(\lambda, p, r').$$

Similar arguments apply to show that $m(\lambda, r', q) \leq m(\lambda, r, q)$.
Therefore, we conclude that

$$m(\lambda, p, q) \leq m(\lambda, p, r') + m(\lambda, r', q) \leq (\lambda, p, r) + m(\lambda, r, p).$$

Hence, **(MM3)** is satisfied.

It must be pointed out that, given a modular metric space (X, w) and fixed $t \in]0, \infty[$, the function $w_t : X \times X \to [0, \infty]$ is not a distance (metric) in general. However, it is not hard to check that m_t is a distance (metric) on D^n for all $t \in]0, \infty[$.

References

1. Beltran-Perez, C., Wei, H.L., Rubio-Solis, A.: Generalized Multiscale RBF Networks and the DCT for Breast Cancer Detection. International Journal of Automation and Computing **17**(1), 55–70 (2020)
2. Chen, Z., Huang, F., Sun, W., Gu, J., Yao, B.: RBF-Neural-Network-Based Adaptive Robust Control for Nonlinear Bilateral Teleoperation Manipulators With Uncertainty and Time Delay. IEEE/ASME Transactions on Mechatronics **25**(2), 906–918 (2020)
3. Chistyakov, V.V.: Modular metric spaces, I: Basic concepts. Nonlinear Analysis: Theory, Methods & Applications **72**(1), 1–14 (2010)
4. Er, M.J., Wu, S., Lu, J., Toh, H.L.: Face recognition with radial basis function (RBF) neural networks. IEEE Transactions on Neural Networks **13**(3), 697–710 (2002)
5. Ferreira Cruz, D.P., Dourado Maia, R., da Silva, L.A., de Castro, L.N.: BeeRBF: A bee-inspired data clustering approach to design RBF neural network classifiers. Neurocomputing **172**, 427–437 (2016)
6. Han, H.G., Ma, M.L., Yang, H.Y., Qiao, J.F.: Self-Organizing Radial Basis Function Neural Network Using Accelerated Second-Order Learning Algorithm. Neurocomputing **469**, 1–12 (2022)
7. Han, H.G., Qiao, J.F.: A structure optimisation algorithm for feedforward neural network construction. Neurocomputing **99**, 347–357 (2013)
8. Han, H.G., Wang, L.D., Qiao, J.F.: Efficient self-organizing multilayer neural network for nonlinear system modeling. Neural Networks **43**, 22–32 (2013)
9. Han, H.G., Zhang, L., Hou, Y., Qiao, J.F.: Nonlinear Model Predictive Control Based on a Self-Organizing Recurrent Neural Network. IEEE Transactions on Neural Networks and Learning Systems **27**(2), 402–415 (2016)
10. Hassoun, M.H.: Fundamentals of Artificial Neural Networks. PHI Learning (2009)
11. Henneron, T., Pierquin, A., Clénet, S.: Surrogate Model Based on the POD Combined with the RBF Interpolation of Nonlinear Magnetostatic FE Model. IEEE Transactions on Magnetics **56**(1), 1–4 (2020)
12. Huang, G.B., Saratchandran, P., Sundararajan, N.: An Efficient Sequential Learning Algorithm for Growing and Pruning RBF (GAP-RBF) Networks. IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics **34**(6), 2284–2292 (2004)
13. Huang, K.Y., Shen, L.C., Weng, L.S.: Radial Basis Function Network for Well Log Data Inversion. In: Proc. International Joint Conference on Neural Networks. pp. 1093–1098 (2011)
14. Keller, J.M., Liu, D., Fogel, D.B.: Fundamentals of Computational Intelligence: Neural Networks, Fuzzy Systems, and Evolutionary Computation. John Wiley & Sons - IEEE Press (2016)
15. Klement, E.P., Mesiar, R., Endre, P.: Triangular Norms. Springer, 1st edn. (2000)

16. Kohonen, T.: Self-Organizing Maps. No. 30 in Series in Information Sciences, Springer, 3rd edn. (2001)
17. Liu, T., Chen, S., Liang, S., Gan, S., Harris, C.J.: Fast Adaptive Gradient RBF Networks for Online Learning of Nonstationary Time Series. IEEE Transactions on Signal Processing **68**, 2015–2030 (2020)
18. Miñana, J.J., Valero, O.: On Indistinguishability Operators, Fuzzy Metrics and Modular Metrics. Axioms **6**(4), 34:1–18 (2017)
19. Moody, J., Darken, C.J.: Fast Learning in Networks of Locally-Tuned Processing Units. Neural Computation **1**(2), 281–294 (1989)
20. Nieto, J.J., Torres, A., Vázquez-Trasande, M.: A Metric Space to Study Differences between Polynucleotides. Applied Mathematics Letters **16**(8), 1289–1294 (2003)
21. Randolph-Gips, M.M., Karayiannis, N.B.: Cosine Radial Basis Function Neural Networks. In: Proc. IEEE International Joint Conference on Neural Networks. vol. 1, pp. 96–101 (2003)
22. Shen, C., Cao, G.Y., Zhu, X.J.: Nonlinear modeling of MCFC stack based on RBF neural networks identification. Simulation Modelling Practice and Theory **10**(1), 109–119 (2002)
23. Shi, X., Cheng, Y., Yin, C., Huang, X., ming Zhong, S.: Design of Adaptive Backstepping Dynamic Surface Control Method With RBF Neural Network for Uncertain Nonlinear System. Neurocomputing **330**, 490–503 (2019)
24. Theodoridis, S., Koutroumbas, K.: Pattern Recognition. Academic Press, USA, 4th edn. (2009)
25. Varvak, M.S.: Pattern Classification Using Radial Basis Function Neural Networks Enhanced with the Rvachev Function Method. In: Proc. Iberoamerican Congress Conference on Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications. pp. 272–279. Springer-Verlag, Berlin, Heidelberg (2011)
26. Wang, S., Yao, X.: Multiclass imbalance problems: Analysis and potential solutions. IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics **42**(4), 1119–1130 (2012)
27. Wong, Y.W., Seng, K.P., Ang, L.M.: Radial Basis Function Neural Network with Incremental Learning for Face Recognition. IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics **41**(4), 940–949 (2011)
28. Wu, J.C., Shen, J., Xu, M., Liu, F.S.: An Evolutionary Self-organizing Cost-Sensitive Radial Basis Function Neural Network to deal with Imbalanced Data in Medical Diagnosis. International Journal of Computational Intelligence Systems **13**(1), 1608–1618 (2020)
29. Xie, Y., Yu, J., Xie, S., Huang, T., Gui, W.: On-line prediction of ferrous ion concentration in goethite process based on self-adjusting structure RBF neural network. Neural Networks **116**, 1–10 (2019)
30. Xiong, T., Bao, Y., Hu, Z., Chiong, R.: Forecasting interval time series using a fully complex-valued RBF neural network with DPSO and PSO algorithms. Information Sciences **305**, 77–92 (2015)
31. Zhang, R., Huang, G.B., Sundararajan, N., Saratchandran, P.: Improved GAP-RBF network for classification problems. Neurocomputing **70**(16), 3011–3018 (2007)