A Modified Hybrid RBF-BP Network Classifier for Nonlinear Estimation/Classification and Its Applications[⋆]

Po-Chai WONG and Jeff Chak-Fu WONG

Department of Mathematics, Chinese University of Hong Kong, Shatin, Hong Kong
`{pcwong,jwong}@math.cuhk.edu.hk`
https://www.math.cuhk.edu.hk/∼jwong/

Abstract. In this work, a modified hybrid radial basis function-back-propagation (RBF-BP) supervised neural network classifier based on the works of Wen et al. [11, 12] is proposed. The modified hybrid RBF-BP network is formulated as an adaptive incremental learning algorithm for a single-layer RBF hidden neuron layer. The algorithm uses a density clustering approach to determine the number of RBF hidden neurons and it maintains the self-learning process of updating the neural network's weights using back-propagation. For the last step of the BP neural network in the modified hybrid classifier, the centers and the width parameters of the basis functions are iteratively updated by the stochastic gradient descent algorithm. As a comparative study, some artificial and real-life datasets, for example, Double Moon, Concentric Circle, No Structure and UCI datasets, are used to test the effectiveness of our homemade implementation strategies. The experimental results showed that the implemented algorithm has significant accuracy improvement and reliability.

Keywords: Radial basis function · Back-propagation · Adaptive hybrid algorithm · Classification.

1 Introduction

A combination of two neural network models, the radial basis function network (RBFN) and the back-propagation network (BPN) (e.g., [5, 7, 2]), are most widely used in nonlinear estimation/classification. RBFN is a local approximation network and the main advantage of the RBFN is that it has only one hidden layer that uses RBF as the activation function. In addition, RBFN maps nonlinearly separable problems in low-dimensional spaces to high-dimensional spaces through RBFs, making them linearly separable in high-dimensional spaces. Its disadvantage is that the classification is slow in comparison to BPN since every node in the hidden layer must compute the RBF function for the input during

⋆ Supported by Department of Mathematics at CUHK and the Teaching Development and Language Enhancement Grant 2022-25 at CUHK.

the classification. Another problem is the number of RBF units. Too many RBF units may result in over-fitting while too few may lead to under-fitting. This number has been manually selected on a trial-and-error basis. Some attempts have been made to decide this number adaptively [11, 12]. BPN is a global approximation neural network. During the training process, the error is propagated backward layer by layer to the input layer, and the ownership value and threshold appearing in the network are corrected. For each training sample, there is only a small number of weights and thresholds to be updated. Other hybrid classifiers, for example, the radial basis function-extreme learning machine classifier for a mixed data type and medical prediction, [13, 6, 10], perform better than the BP classifier. Further extensions of the hybrid RBF-BP network classifier (the Hybrid classifier for short) using pre-RBF kernels were found in [14, 15].

In this paper, a modified version of the hybrid classifier (the mHybrid classifier for short) is proposed based on the works of Wen et al. [11, 12]. It is formulated as an adaptive incremental learning algorithm for a single-layer RBF hidden neuron layer by fixing/shifting the center, adjusting the width parameter of the basis functions and updating the number of RBF hidden neurons, and it maintains a self-learning process of tuning a single-layer BP neural network's weights to improve classification accuracy. In the mHybrid classifier, the center and the width parameter of the RBFs are iteratively updated by the stochastic gradient descent (SGD) algorithm.

The rest part of the paper is outlined as follows. Section 2 presents the architecture of the mHybrid network and the centers, widths and number of RBF hidden neurons interacting with the multi-layer perceptron (MLP) hidden neurons. In Algorithm 1, with optimally determined centers and width parameters, the coverage effect of each hidden neuron can be guaranteed. In Section 3, by passing the output of the RBF hidden neurons into an MLP neural network, backpropagation (BP) is used to update the weights of the MLP. Bridging between RBF-BP networks is shown in Algorithm 2 and their implementations are highlighted. In Algorithm 3, two SGD iterative steps are proposed for updating the centers and the width parameters of the RBF units. Section 4 examines the numerical performance of the mHybrid classifier on artificial datasets, e.g., Double Moon, Concentric Circle (e.g., [4]). No Structure [9] and real-life UCI datasets [3]. Section 5 summarizes our findings and concludes the paper.

2 Structure of the Incremental Learning Algorithm

This section describes the three sequential steps used to first find the centers for the RBFs, then find the widths for the RBFs, and to determine a new center as needed.

2.1 Finding the centers for the radial basis functions

Our aim here is to introduce an efficient technique of density clustering for balanced data. For the RBF networks, the data $\{\boldsymbol{x}\}$ of each class y is covered by circles of different sizes. To decide the optimal number of circles, a pre-selected discriminant function is designed. Then the locations and the widths of the circles are determined from the repulsive force exerted by nearby heterogeneous members, so that each circle contains many homogeneous members and few heterogeneous members. Formally, we define a set of discriminant functions ρ_i, one for each class i [1],

$$\rho_i(\boldsymbol{x}) > \rho_j(\boldsymbol{x}) \text{ for any } j \neq i \implies \boldsymbol{x} \text{ belongs to class } i. \tag{1}$$

The higher the value of ρ_i, the more likely that $\boldsymbol{x}$ belongs to class i. Another natural assumption is that the more members of the same class there are around $\boldsymbol{x}$, the more likely $\boldsymbol{x}$ belongs to the same class. To obtain the discriminant function, one therefore can define a potential function γ as

$$\gamma(\boldsymbol{x}, \boldsymbol{z}) = \frac{1}{1 + T\|\boldsymbol{x} - \boldsymbol{z}\|^2}, \tag{2}$$

where $T > 0$ is a distance weighting factor and $\|\cdot\|$ is the Euclidean norm. The potential γ is a particular example of the general inverse multi-quadratic function, $(1 + \epsilon^2\|\boldsymbol{x} - \boldsymbol{z}\|^2)^{-p/2}$ when $T = \epsilon^2$ and $p = 2$. The potential γ is proportional to the closeness between two points $\boldsymbol{x}, \boldsymbol{y}$.

Given a data set S that consists of N training samples $\{(\boldsymbol{x}_k, y_k)\}_{k=1}^N$, where $\boldsymbol{x}_k \in \mathbb{R}^n$ and $y_k \in \mathbb{R}^H$, where n is the dimension of $\boldsymbol{x}_k$ and H is the dimension of y_k. Let $S^i = \{\boldsymbol{x}_k^i : \ k = 1, \cdots, N_i\}$ be the set of training samples in the ith pattern class, N_i the number of samples in class i and $S^i \cap S^j = \emptyset$ for $i \neq j$. For $\boldsymbol{x} \in \mathbb{R}^n$ and $\boldsymbol{x}_k^i \in S^i$, a discriminant function ρ_i can be constructed by the superposition of such potential functions $\gamma(\cdot, \boldsymbol{x}_k^i)$:

$$\rho_i(\boldsymbol{x}) = \sum_{k=1}^{N_i} \gamma(\boldsymbol{x}, \boldsymbol{x}_k^i) = \sum_{k=1}^{N_i} \frac{1}{1 + T\|\boldsymbol{x} - \boldsymbol{x}_k^i\|^2}. \tag{3}$$

As shown in Figure 1, $T > 0$ controls the width of the region of influence and the sharpness of the distribution, e.g., the larger the factor T, the sharper ρ_i and the distribution of the samples will have a better shape.

It is worth mentioning that the ρ_i defined here is different from that in Wen et. al. [11, 12], where when evaluated at sample point $\boldsymbol{x}_l^i$, the summation does not consider that point (i.e., $\widetilde{\rho}_i(\boldsymbol{x}_l^i) = \sum_{k=1,k\neq l}^{N_i} \gamma(\boldsymbol{x}_l^i, \boldsymbol{x}_k^i)$). Then we have $\rho_i(x) = \widetilde{\rho}_i(x) + 1$ for $x^i \in S^i$. This difference is not critical in the end (except for the choice of δ) since we will only compare the values within the same class. Here are a few properties of γ [8]:

1. $\gamma(\boldsymbol{x}, \boldsymbol{z})$ attains maximum at $\boldsymbol{x} = \boldsymbol{z}$.
2. $\gamma(\boldsymbol{x}, \boldsymbol{z})$ tends to 0 as $\|\boldsymbol{x} - \boldsymbol{z}\|$ increases.
3. $\gamma(\boldsymbol{x}, \boldsymbol{z})$ is smooth and decreases monotonically as $\|\boldsymbol{x} - \boldsymbol{z}\|$ increases.
4. $\gamma(\boldsymbol{x}_1, \boldsymbol{z}) = \gamma(\boldsymbol{x}_2, \boldsymbol{z})$ if $\|\boldsymbol{x}_1 - \boldsymbol{z}\| = \|\boldsymbol{x}_2 - \boldsymbol{z}\|$.

These properties allow ρ_i to capture the local influence of S^i at $\boldsymbol{x}$ and to correlate to the likeliness for the input $\boldsymbol{x}$ to belong to S^i. Finally, all that remains to be checked is whether this ρ_i can classify correctly, as stated in the theorem below.

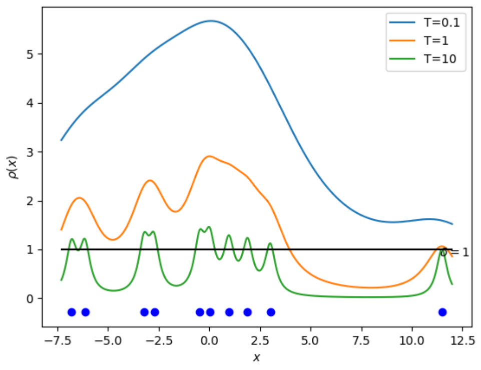


Fig. 1: Illustration of the effect of T in a 1D example using 10 data samples.

Theorem 1. Suppose $\boldsymbol{x}_l^i \in S^i := \{\boldsymbol{x}_1^i, \cdots, \boldsymbol{x}_{N_i}^i\}$. Let N be the number of classes. Let $D = \min\{\|\boldsymbol{x}^i - \boldsymbol{x}^j\|^2 : \boldsymbol{x}^i \in S^i, \boldsymbol{x}^j \in S^j \text{ for any } i \neq j,\ i, j \leq N\} > 0$. Then, there exists a continuously differentiable discriminant function ρ_i such that

$$\rho_i(\boldsymbol{x}_l^i) > \rho_j(\boldsymbol{x}_l^i) \text{ for any } j \neq i.$$

Proof. Suppose $S^j = \{\boldsymbol{x}_1^j, \cdots, \boldsymbol{x}_{N_j}^j\}$. Let $T > \frac{1}{D}(N_j - 1)$. Define ρ_i as stated before using this T. Then,

$$\begin{aligned}
\rho_j(\boldsymbol{x}_l^i) &= \sum_{k=1}^{N_j} \frac{1}{1 + T\|\boldsymbol{x}_l^i - \boldsymbol{x}_k^j\|^2} \leq N_j \max_{k=1,\cdots,N_j} \frac{1}{1 + T\|\boldsymbol{x}_l^i - \boldsymbol{x}_k^j\|^2} \\
&\leq N_j \frac{1}{1 + T \min_{k=1,\cdots,N_j} \|\boldsymbol{x}_l^i - \boldsymbol{x}_k^j\|^2} \leq \frac{N_j}{1 + TD} < 1 \\
&= \frac{1}{1 + T\|\boldsymbol{x}_l^i - \boldsymbol{x}_l^i\|^2} \leq \sum_{k=1}^{N_i} \frac{1}{1 + T\|\boldsymbol{x}_l^i - \boldsymbol{x}_k^i\|^2} = \rho_i(\boldsymbol{x}_l^i).
\end{aligned}$$

□

The value T can be selected as $T > \frac{1}{D}(N_j - 1)$ for any j such that the theorem holds.

Now, using the discriminant function ρ_i, we can select an initial center by taking the maximum of the discriminant function over class i since the samples are concentrated/localized around it. Define the initial center

$$\boldsymbol{\mu}_k := \arg\max\{\rho_i(\boldsymbol{x}_l^i) :\ \boldsymbol{x}_l^i \in S^i\}. \tag{4}$$

We now circle it with a given predetermined radius σ with a predefined threshold $\sigma_{\min} < \sigma$. Our next goal is to move the center $\boldsymbol{\mu}_k$ and resize the circle so as to reduce the number of heterogeneous members around $\boldsymbol{\mu}_k$.

A repulsive force $\boldsymbol{F}$ from each heterogeneous member $\boldsymbol{x}^j$ in the ball $\mathcal{B}(\boldsymbol{\mu}_k, \lambda\sigma) := \{\boldsymbol{x} : \ \|\boldsymbol{x} - \boldsymbol{\mu}_k\| \leq \lambda\sigma\}$, where $\lambda > 1$ is a width covering factor, is given by

$$\boldsymbol{F} = \exp\left(-\alpha\|\boldsymbol{\mu}_k - \boldsymbol{x}^j\|\right) \frac{\boldsymbol{x}^j - \boldsymbol{\mu}_k}{\|\boldsymbol{x}^j - \boldsymbol{\mu}_k\|}, \tag{5}$$

where α is the repulsive force control factor. The exponential scale factor term is used to control the variation of the center drift due to any heterogeneous members within the ball.

The center position is updated to move away from heterogeneous members as follows:

$$\boldsymbol{\mu}_k \leftarrow \boldsymbol{\mu}_k + \boldsymbol{F}. \tag{6}$$

The preassigned value $\alpha > 0$ should not be too small otherwise it loses the purpose of maximizing the discriminant function ρ_i. But a very small α may not reduce the number of heterogeneous members in the ball. A big center shift may also result in covering new heterogeneous members. Therefore, this choice of α is highly dependent on the dataset. In some cases, center drifts are not sufficient to reduce the number of heterogeneous members. We, therefore, fix the maximum number of iterations Epo allowed for testing for center drifts.

We count the number $M_{\neq i}$ of heterogeneous samples $\boldsymbol{x} \in S \backslash S^i$ in the current ball $\mathcal{B}(\boldsymbol{\mu}_k, \lambda\sigma)$ by $\|\boldsymbol{x} - \boldsymbol{\mu}\| \leq \lambda\sigma$. Define M_i as the number of samples in S^i in $\mathcal{B}(\boldsymbol{\mu}_k, \lambda\sigma)$. The center for each ball can be adjusted by the sum of the resultant forces:

$$\boldsymbol{\mu}_k \leftarrow \boldsymbol{\mu}_k + \frac{1}{M} \sum_{p=1}^{M_{\neq i}} \exp\left(-\alpha\|\boldsymbol{\mu}_k - \boldsymbol{x}_p\|\right) \frac{\boldsymbol{x}_p - \boldsymbol{\mu}_k}{\|\boldsymbol{x}_p - \boldsymbol{\mu}_k\|}, \tag{7}$$

where M is the preassigned value to average all the resultant forces.

After an iteration, check the number of heterogeneous members in the ball again. If the new number $M'_{\neq i}$ of heterogeneous members in the ball is reduced, the iterative process continues, otherwise the center is fixed. Once the center is positioned, the algorithm readjusts the size of the ball to mostly cover only homogeneous members. An RBF center is then selected.

2.2 Finding the widths for the radial basis functions

If heterogeneous members exist in the ball (i.e., $M'_{\neq i} > 0$), we shrink the ball so that it almost covers the closest heterogeneous member $\boldsymbol{x}^j$ from $\boldsymbol{\mu}_k$ by the formula below. To avoid over-shrinking, $\beta > 1$ acts as a relaxation factor and $\sigma_{\min}$ as the lower bound of the size.

$$\sigma_k \leftarrow \begin{cases} \max\{\min\{\|\boldsymbol{\mu}_k - \boldsymbol{x}\|/\beta : \ \boldsymbol{x} \in S\backslash S^i\}, \sigma_{\min}\} & \text{if } M'_{\neq i} > 0 \\ \sigma & \text{if } M'_{\neq i} = 0 \end{cases}. \tag{8}$$

This β should be slightly smaller than λ to avoid setting too large a value on λ. Suppose λ/β is less than but close to 1. Then for any $\boldsymbol{x}^{\neq i} \in (S\backslash S^i) \cap \mathcal{B}(\boldsymbol{\mu}_k, \lambda\sigma)$,

$$\sigma_{\min} \leq \sigma_k \leq \min\{\|\boldsymbol{\mu}_k - \boldsymbol{x}\|/\beta : \ \boldsymbol{x} \in S\backslash S^i\} \leq \|\boldsymbol{\mu}_k - \boldsymbol{x}^{\neq i}\|/\beta \leq (\lambda/\beta)\sigma \leq \sigma. \tag{9}$$

Since the last inequality is "slightly less than", if $M'_{\neq i} > 0$, the updated circle is not too large when controlled by β.

2.3 Updating the discriminant function

To decide if a new center is needed, the discriminant function ρ_i is updated to remove the influence of the centers found so far. The processes of shifting the center and resizing the RBF balls repeat if some updated potential is above the threshold δ, i.e.,

$$\max\{\rho_i^{\text{new}}(\boldsymbol{x}_1^i), \cdots, \rho_i^{\text{new}}(\boldsymbol{x}_{N_i})\} > \delta, \tag{10}$$

where

$$\rho_i^{\text{new}}(\boldsymbol{x}) = \rho_i(\boldsymbol{x}) - \rho_i(\boldsymbol{\mu}_k) \exp\left(-\frac{1}{2\sigma_l^2}\|\boldsymbol{x} - \boldsymbol{\mu}_k\|\right). \tag{11}$$

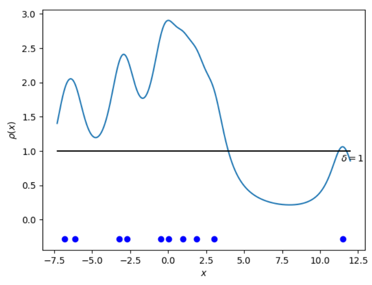
If all of the new ρ_i^{new} are less than δ, the remaining data are not dense enough to form an effective cluster center. It is important to note that since all the terms in the discriminant function ρ_i are positive, this δ is dependent on the size N_i and the scaling parameter T, as shown in Figure 1, and thus can only be determined on an empirical basis. For the overall effect of δ, see Figures (2a) - (2d). An alternative method is to normalize a new discriminant function $\widetilde{\rho}_i := \frac{1}{N_i}\sum_k \gamma(\cdot, \boldsymbol{x}_k^i)$. The T in the proof of Theorem 1 is then dependent on N_i. Future studies can then be done on the suggested selection of δ for every class i.

The whole center selection process is summarised in Algorithm 1.

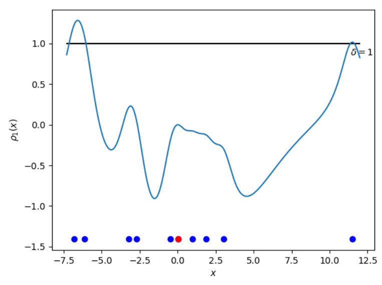
3 Moving from the RBFN to the BPN

The algorithm follows the original purpose of imposing the RBF layer, which captures the relation of the classes with the points of high relative density. However, it is natural to question the relation of such a doctrine with the ultimate purpose of reducing classification error.

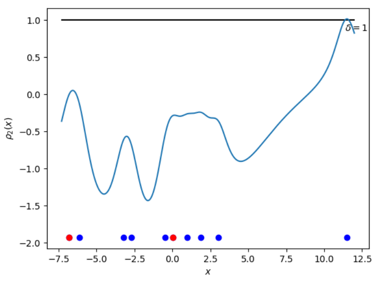
In MLP, a gradient-search algorithm is applied to iteratively update the weights using backpropagation in order to attain a local minimum of the error function. This simple idea comes from the fact that the infimum of a normed space is not bigger than the infimum of a subspace. The motivation for our extension comes from questioning whether extending the parameter space in the gradient-search algorithm would improve the classification. By combining the hybrid structure, we apply backpropagation to both MLP and RBF layers with the initial guess of RBF centers selected by the incremental algorithm above.



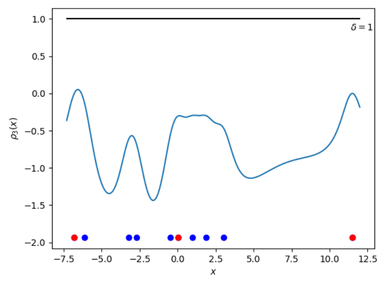
(a) Step 1



(b) Step 2



(c) Step 3



(d) Step 4

Fig. 2: This shows an example of 10 homogeneous data points (blue). The black horizontal line marks the threshold $\delta = 1$. The blue curve is the value of $\rho(x)$ with $T = 1$. We pick the data point with the highest value of ρ in Figure (2a) and mark it as our first cluster point $\boldsymbol{\mu}_1$ (red). (2b) The function ρ_1 is updated, penalizing the region around $\boldsymbol{\mu}_1$. Since some points are above the threshold, the selection continues. (2c) $\boldsymbol{\mu}_2$ is found and marked in red. Note that this showcases the importance of δ since the data point almost does not pass the test. (2d) $\boldsymbol{\mu}_3$ is found. The graph lies below the threshold and the selection process terminates.

The main concern regarding this approach is the problem of hitting local minima. Here, we mainly focus on the extent to which the classifier gets stuck at the local minima after the extension of the parameter space. Therefore, we use the common stochastic gradient descent method.

Let K be the number of RBF centers. Define the output of RBF at each center $\boldsymbol{\mu}_k$ by

$$\phi_k(\boldsymbol{x}; \boldsymbol{\mu}_k, \sigma_k) = \exp\left(-\frac{1}{2\sigma_k^2}\|\boldsymbol{x} - \boldsymbol{\mu}_k\|^2\right) \text{ for } k = 1, 2, \cdots, K. \tag{12}$$

The output of $\varPhi(\boldsymbol{x}) := [\phi_1(\boldsymbol{x}; \boldsymbol{\mu}_1, \sigma_1), \cdots, \phi_K(\boldsymbol{x}; \boldsymbol{\mu}_K, \sigma_K)]$ is nonnegative. To facilitate computational speed, we polarise them by a mapping $\boldsymbol{x} \mapsto 2\boldsymbol{x} - 1$, which becomes the input of a feedforward network with the activation function

Algorithm 1: Incremental Learning Algorithm for Constructing RBF Hidden Neurons

```
Data: $S = \cup_{i=1}^{H} S^i$, where $S^i = \{\boldsymbol{x}_1^i, \cdots, \boldsymbol{x}_{N_i}^i\}$ the set of training samples
      labeled $i$ and $H$ is the total number of classes
Result: $\{\boldsymbol{\mu}_1, \cdots, \boldsymbol{\mu}_K\}$, $\{\sigma_1, \cdots, \sigma_K\}$
let
    $\sigma > \sigma_{\min} > 0, k \leftarrow 0, T, M, \alpha, \beta, \lambda, \mathsf{Epo} > 0$
for $i = 1$ to $H$ do
    Define the discriminant function $\rho_i$ for class $i$ by $\rho_i(\boldsymbol{x}) := \sum_{k=1}^{N_i} \gamma(\boldsymbol{x}, \boldsymbol{x}_k^i)$
    repeat
        $\boldsymbol{\mu}_k \leftarrow \arg\max\{\rho_i(\boldsymbol{x}_k^i) : \ \boldsymbol{x}_k^i \in S^i\}$
        Define $M_i$ and $M_{\neq i}$
        for each sample $\boldsymbol{x}_p \in S \backslash S^i$ in $\mathcal{B}(\boldsymbol{\mu}_k, \lambda\sigma)$ do
            $\boldsymbol{\mu}_k \leftarrow \boldsymbol{\mu}_k + \exp(-\alpha \|\boldsymbol{\mu}_k - \boldsymbol{x}_p\|) \frac{\boldsymbol{x}_p - \boldsymbol{\mu}_k}{\|\boldsymbol{x}_p - \boldsymbol{\mu}_k\|}$
        end
        Define the updated $M_i'$ and $M_{\neq i}'$
        $m \leftarrow 0$
        while $M_{\neq i}' > 0$ and $m \leq \mathsf{Epo}$ do
            if $M_i' \geq M_i$ and $M_{\neq i}' \leq M_{\neq i}$ then
                Update with $M_i \leftarrow M_i'$; $M_{\neq i} \leftarrow M_{\neq i}'$
                $\mu_k \leftarrow \mu_k + \frac{1}{M} \sum_{p=1}^{M_{\neq i}} \exp(-\alpha \|\boldsymbol{\mu}_k - \boldsymbol{x}_p\|) \frac{\boldsymbol{x}_p - \boldsymbol{\mu}_k}{\|\boldsymbol{x}_p - \boldsymbol{\mu}_k\|}$
                $m \leftarrow m + 1$
            else
                $\sigma_k \leftarrow \begin{cases} \max\{\min\{\|\boldsymbol{\mu}_k - \boldsymbol{x}\|/\beta : \boldsymbol{x} \in S \backslash S^i\}, \sigma_{\min}\} & \text{if } M_{\neq i}' > 0 \\ \sigma_{\min} & \text{if } M_{\neq i}' = 0 \end{cases}$
            end
        end
        Define $\rho_i^{\text{new}}(\boldsymbol{x}) := \rho_i(\boldsymbol{x}) - \rho_i(\boldsymbol{\mu}_k) \exp\left(-\frac{1}{2\sigma_k^2} \|\boldsymbol{x} - \boldsymbol{\mu}_k\|\right)$
        $\rho_i \leftarrow \rho_i^{\text{new}}$
        $k \leftarrow k + 1$
    until
        $\max\left\{\rho_i(\boldsymbol{x}_1^i), \cdots, \rho_i(\boldsymbol{x}_{N_i}^i)\right\} \leq \delta$
end
```

$\boldsymbol{x} \mapsto a \tanh(b\boldsymbol{x})$, where a and b are constants. They are updated using back-propagation, passing the error term in the output to each hidden neuron.

In Algorithm 2, $\mathbf{W} = [W^{(1)}, \cdots, W^{(L)}]$ is a tensor with each $W^{(l)}$ being the weight matrix between the $(l-1)$- and l-layers, where L is the number of BP hidden layers (including the output layer). $\boldsymbol{\delta}^{(l)} := [\delta_1^{(l)}, \cdots, \delta_{n_l}^{(l)}]$ is a vector recording the error passed to the n_l hidden neurons of the l-th layer, $l = 1, \cdots, L$. $\boldsymbol{\delta}^{(0)}$ is the error vector for the RBF units. The error term of the first BP layer $\boldsymbol{\delta}^{(1)}$ can further be passed to the RBF layer, since $\boldsymbol{x} \mapsto \Phi(\boldsymbol{x})$ is smooth with respect to $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ (for strictly positive $\boldsymbol{\sigma}$). Suppose $e = \sqrt{\sum_{j=1}^{H} (y_j - \widetilde{y}_j^{(L)})^2}$ is the error of our predicted output $\widetilde{\boldsymbol{y}}^{(L)}$ of the L-th layer. By the Chain rule, we

Algorithm 2: Hybrid RBF-BP Network Architecture

```
   Data: $S = \cup_{i=1}^{H} S^i$, where $S^i = \{\boldsymbol{x}_1^i, \cdots, \boldsymbol{x}_{N_i}^i\}$, and Initialized weights $\mathbf{W}$
   Result: Updated weights $\mathbf{W}$
1  Define constants $a, b$, learning rate $\eta$
2  Apply Algorithm 1 to obtain centers $\{\boldsymbol{\mu}_1, \cdots, \boldsymbol{\mu}_K\}$ and widths $\{\sigma_1, \cdots, \sigma_K\}$
3  repeat
4      $\widetilde{\boldsymbol{y}}^{(\mathbf{0})} \leftarrow 2\Phi(\boldsymbol{x})^T - 1$
5      for $l = 1$ to $L$ do
6          $\boldsymbol{v}^{(l)} \leftarrow W^{(l)} \widetilde{\boldsymbol{y}}^{(l-1)}$
7          $\widetilde{\boldsymbol{y}}^{(l)} \leftarrow a \tanh(b\boldsymbol{v}^{(l)})$
8      end
9      $\boldsymbol{E} \leftarrow$ the error term $(\boldsymbol{y} - \widetilde{\boldsymbol{y}}^{(L)})$
10     for $l = L$ to $l = 1$ do
11         if $l = L$ then
12             $\boldsymbol{\delta}^{(L)} \leftarrow ab\boldsymbol{E} \odot \operatorname{sech}^2(b\boldsymbol{v}^{(L)})$      $\odot$ is elementwise multiplication
13         else
14             $\boldsymbol{\delta}^{(l)} \leftarrow ab \operatorname{sech}^2(b\boldsymbol{v}^{(l)}) \odot \left(W^{(l+1)} \boldsymbol{\delta}^{(l+1)}\right)$
15         end
16         $W^{(l)} \leftarrow W^{(l)} + \eta \boldsymbol{\delta}^{(l)} (\widetilde{\boldsymbol{y}}^{(l-1)})^T$
17     end
18 until
19     $e <$ Tolerance
```

obtain

$$\begin{cases} \dfrac{\partial e}{\partial \boldsymbol{\mu}_k} = \dfrac{\partial e}{\partial \widetilde{\boldsymbol{y}}_k^{(0)}} \dfrac{\partial \widetilde{\boldsymbol{y}}_k^{(0)}}{\partial \boldsymbol{\mu}_k} = \delta_k^{(0)} \dfrac{\partial \widetilde{\boldsymbol{y}}_k^{(0)}}{\partial \boldsymbol{\mu}_k} = \delta_k^{(0)} \dfrac{\partial}{\partial \boldsymbol{\mu}_k} \phi_k(\boldsymbol{x}; \boldsymbol{\mu}_k, \sigma_k) \\ \qquad = \delta_k^{(0)} \exp\left(\dfrac{-\|\boldsymbol{x} - \boldsymbol{\mu}_k\|^2}{2\sigma_k^2}\right) \dfrac{1}{\sigma_k^2} (\boldsymbol{x} - \boldsymbol{\mu}_k) \\ \dfrac{\partial e}{\partial \sigma_k} = \dfrac{\partial e}{\partial \widetilde{\boldsymbol{y}}_k^{(0)}} \dfrac{\partial \widetilde{\boldsymbol{y}}_k^{(0)}}{\partial \sigma_k} = \delta_k^{(0)} \dfrac{\partial \widetilde{\boldsymbol{y}}_k^{(0)}}{\partial \sigma_k} = \delta_k^{(0)} \dfrac{\partial}{\partial \sigma_k} \phi_k(\boldsymbol{x}; \boldsymbol{\mu}_k, \sigma_k) \\ \qquad = \delta_k^{(0)} \exp\left(\dfrac{-\|\boldsymbol{x} - \boldsymbol{\mu}_k\|^2}{2\sigma_k^2}\right) \dfrac{1}{\sigma_k^3} \|\boldsymbol{x} - \boldsymbol{\mu}_k\|^2 \end{cases} . \tag{13}$$

Then, the update rules of $\boldsymbol{\mu}_k$ and σ_k follow from SGD with learning rate η:

$$\boldsymbol{\mu}_k \leftarrow \boldsymbol{\mu}_k + 2\eta\delta_k^{(0)} \frac{\partial \widetilde{\boldsymbol{y}}^{(0)}}{\partial \boldsymbol{\mu}_k}(\boldsymbol{x}) \quad \text{and} \quad \sigma_k \leftarrow \sigma_k + 2\eta\delta_k^{(0)} \frac{\partial \widetilde{\boldsymbol{y}}^{(0)}}{\partial \sigma_k}(\boldsymbol{x}). \tag{14}$$

Let $\frac{\partial \widetilde{\boldsymbol{y}}^{(\mathbf{0})}}{\partial \boldsymbol{\mu}} = \left[\frac{\partial \widetilde{\boldsymbol{y}}^{(0)}}{\partial \boldsymbol{\mu}_1}, \cdots, \frac{\partial \widetilde{\boldsymbol{y}}^{(0)}}{\partial \boldsymbol{\mu}_K}\right]$ and $\frac{\partial \widetilde{\boldsymbol{y}}^{(\mathbf{0})}}{\partial \boldsymbol{\sigma}} = \left[\frac{\partial \widetilde{\boldsymbol{y}}^{(0)}}{\partial \sigma_1}, \cdots, \frac{\partial \widetilde{\boldsymbol{y}}^{(0)}}{\partial \sigma_K}\right]$. The investigated algorithm is shown as Algorithm 3, where $diag(\boldsymbol{x})$ is the diagonal matrix with the diagonal element being $\boldsymbol{x}$.

Algorithm 3: Modified Hybrid RBF-BP Network Architecture

```
  Data: $S = \cup_{i=1}^{H} S_i$, where $S_i = \{\boldsymbol{x}_1^i, \cdots, \boldsymbol{x}_{N_i}^i\}$ and Initialized weights $\mathbf{W}$
  Result: updated weights $\mathbf{W}$, centers $\{\boldsymbol{\mu}_k\}$ and widths $\{\sigma_k\}$
1 Define constants $a, b$, learning rate $\eta$
2 Apply Algorithm 1 to obtain centers $\boldsymbol{\mu} = [\boldsymbol{\mu}_1, \cdots, \boldsymbol{\mu}_K]$ and widths
    $\boldsymbol{\sigma} = [\sigma_1, \cdots, \sigma_K]$
3 repeat
4     Lines $4 - 17$ in Algorithm 2
5     $\boldsymbol{\delta}^{(0)} \leftarrow W^{(1)} \boldsymbol{\delta}^{(1)}$
6     $\boldsymbol{\mu} \leftarrow \boldsymbol{\mu} + 2\eta \frac{\partial \widetilde{\boldsymbol{y}}^{(0)}}{\partial \boldsymbol{\mu}}(\boldsymbol{x}) diag(\boldsymbol{\delta}^{(0)})$
7     $\boldsymbol{\sigma} \leftarrow \boldsymbol{\sigma} + 2\eta \frac{\partial \widetilde{\boldsymbol{y}}^{(0)}}{\partial \sigma}(\boldsymbol{x}) diag(\boldsymbol{\delta}^{(0)})$
8 until
9     $e <$ Tolerance
```

4 Numerical Results

As a comparative study, some artificial and real-life datasets were used to test the effectiveness of our implementation strategies and our homemade MATLAB codes.

4.1 Comparison of MLP and mHybrid classifiers on different datasets

We used 8 datasets with classes as shown in Table 1 and all the parameters used for the mHybrid classifier are in Table 2. We set Tolerance $= 10^{-6}$. Figure 3 shows the patterns for Double Moon with 2 classes, Concentric Circles (CC), Concentric Circles with 2 extra layers (CC2) and Double Moon with 6 classes (DM6) (Each crescent is split into three sections). For each dataset, we first obtained the theoretical value $\max\{(N_j - 1)/D\}$ (the last column in Table 2), then we adjusted the T value descendingly to obtain the best result. Table 1 shows the results of the MLP and mHybrid classifiers. To assess their performance, accuracy, precision, recall, and F-score were used. Table 1 shows that the mHybrid classifier outperformed the MLP classifier.

We further examined the pendigit dataset with 10 classes. Different network architectures of MLP and mHybrid classifiers are shown in Figure 4, where the first label refers to the number of hidden neurons of the two layers for MLP, while the second label refers to the number of RBF centers and hidden neurons used in the mHybrid classifier. According to the results, the accuracy of the mHybrid classifier is significantly better than that of the MLP classifier when the numbers of RBF centers and hidden neurons increase.

4.2 Comparison of Hybrid and mHybrid classifiers using DM6 and No Structure datasets

In what follows, we numerically analyze the effect of the center and the width parameter of the RBFs, which are iteratively updated by the SGD algorithm,

Table 1: Performance comparison of MLP and mHybrid classifiers on different datasets.

		MLP				**mHybrid**			
Dataset	Classes	Accuracy	Precision	Recall	F Score	Accuracy	Precision	Recall	F Score
Pendigit	10	0.9780	0.8938	0.8906	0.8906	0.9819	0.9126	0.9109	0.9089
Letter	26	0.9781	0.7127	0.7121	0.7070	0.9821	0.8000	0.7660	0.7680
Ozone	2	0.9448	0.7101	0.6135	0.6354	0.9540	0.9259	0.6741	0.7371
Sonar	2	0.8428	0.8533	0.8468	0.8424	0.8809	0.8820	0.8813	0.8807
Iono	2	0.9277	0.9099	0.9357	0.9166	0.9277	0.9282	0.9175	0.9214
DM	2	0.9983	0.9955	0.9960	0.9957	0.9933	0.9790	0.9826	0.9803
CC	2	0.6694	0.6700	0.6691	0.6688	0.9008	0.8992	0.8992	0.8992
CC2	2	0.9240	0.9239	0.9243	0.9240	0.9557	0.9556	0.9564	0.9557

Table 2: Parameters of the mHybrid classifier used on different datasets.

	Parameters										
Dataset	a	b	T	α	β	σ_{min}	σ_{init}	λ	Epo	δ	$\max(N_j-1)/D$
Pendigit	1.2	0.8	50	25	1.2	1e-4	0.5	1.3	10	0.001	68.94
Letter	1.2	0.8	100	25	1.2	1e-4	5	1.3	10	0.001	813
Ozone	1.2	0.8	50	25	1.2	1e-4	5	1.3	10	0.01	281.78
Sonar	1.2	0.8	100	25	1.2	1e-4	5	1.3	10	0.001	216.83
Iono	1.2	0.8	100	25	1.2	1e-4	5	1.3	10	0.001	479.98
DM	1.2	0.8	50	25	1.2	1e-4	0.5	1.3	10	0.001	7.15e+04
CC	1.2	0.8	100	25	1.2	1e-4	0.5	1.3	10	0.001	5.86e+04
CC2	1.2	0.8	200	25	1.2	1e-4	0.5	1.3	10	0.001	6.53e+03
DM6	1.2	0.8	500	25	1.2	1e-4	0.7	1.3	10	0.01	151.65
No Structure	1.2	0.8	200	25	1.2	1e-4	See Figure 6	1.3	10	0.001	9.23e+03

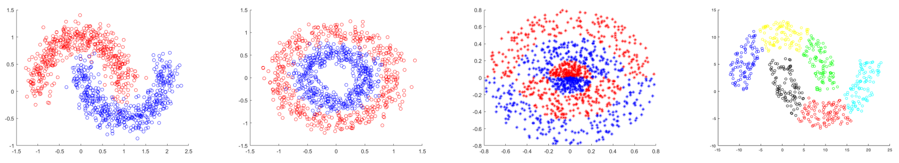


Fig. 3: (3a) Double Moon with 2 classes. (3b) Concentric Circles (CC). (3c) Concentric Circles with 2 extra layers (CC2). (3d) Double Moon with 6 classes (DM6).

referred to as Algorithm 3. Using the DM6 dataset, we observed that the mHybrid classifier outperformed the MLP and Hybrid classifiers as shown in Table 3.

Figure 5 shows the pattern of No Structure datasets, where data is labeled using agglomerative clustering algorithms with different linkages ('Ward', 'Av-

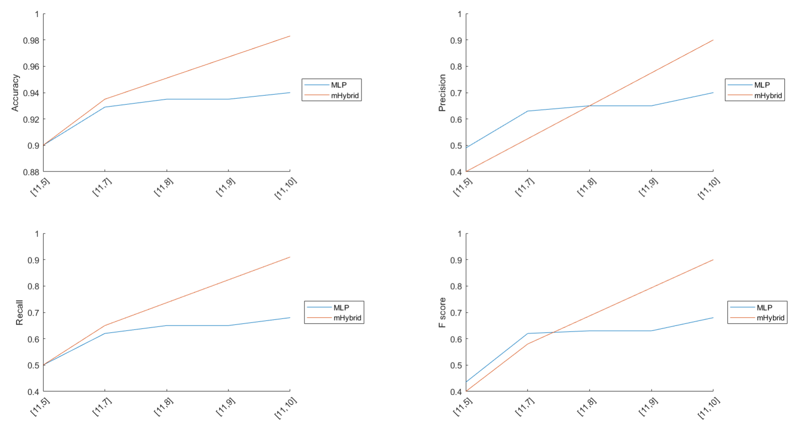


Fig. 4: Comparison of the different network architectures of two classifiers using the pendigit dataset. For MLP, the two numbers are the numbers of hidden neurons of the two hidden layers. For mHybrid, the first number is the number of RBF units and the second is the BP units.

Table 3: Comparison of three classifiers using DM6.

Classifier	Accuracy	Precision	Recall	F Score
Modified Hybrid RBF-BP	0.9933	0.9790	0.9826	0.9803
Hybrid RBF-BP	0.9811	0.9532	0.9624	0.9703
MLP	0.9552	0.8816	0.8849	0.8567

erage', and 'Complete') with different numbers of classes. To be more precise, 'Ward' minimizes the variance of each cluster, 'Average' uses the average of the distances of each sample pair from two clusters, and 'Complete' uses the maximum distances between all samples of the clusters. The datasets are tested on different initial σ values with identical configurations. Figure 6 shows that the mHybrid classifier outperformed the hybrid classifier. In addition to that, the smaller the σ value, the more RBF centers generated, and thus accuracy increased.

5 Conclusions

The performance of the modified hybrid RBF-BP classifier has been tested using artificial and real-life datasets. Based on the numerical results, we concluded that when using the modified classifier, including the lines for updating the last step yielded better accuracy than not including them. Although it performs better than the MLP classifier, higher computational efficiency will be required if more testing is to be done. In the future, tests may be done to fine-tune the parameters to find a set of all optimal parameters for the modified hybrid classifier. Moreover, the efficiency of combining different classifiers with the proposed classifier requires more work.

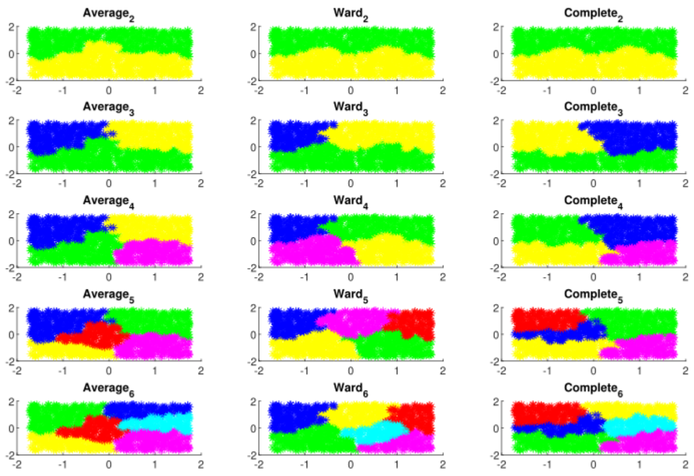


Fig. 5: Distribution of No Structure datasets.

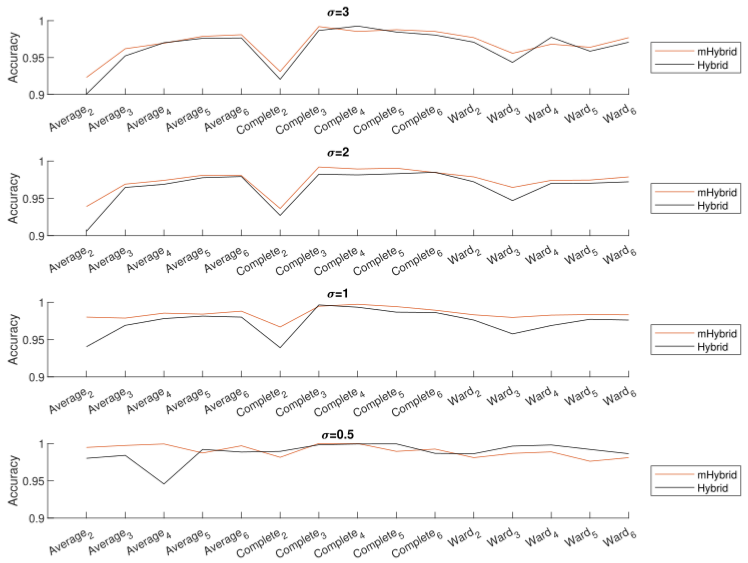


Fig. 6: Comparison of two classifiers on different No Structure datasets.

References

1. Babu, C.C., Kalra, S.N.: On the application of Bashkirov, Braverman, and Muchnik potential function for feature selection in pattern recognition. Proceedings of the IEEE 60(3), 333-334 (1972)
2. Deng, Y., Zhou, X., Shen, J., Xiao, G., Hong, H., Lin, H., Wu, F., Liao, B.Q: New methods based on back propagation (BP) and radial basis function (RBF) artificial neural networks (ANNs) for predicting the occurrence of haloketones in tap water. Sci Total Environ Jun 10;772:145534 (2021) doi: 10.1016/j.scitotenv.2021.145534. Epub 2021 Feb 2. PMID: 33571763.
3. Dua, D., Graff, C.: UCI Machine Learning Repository (2017) Available from: http://archive.ics.uci.edu/ml
4. Haykin, S. Neural Networks and Learning Machines. Third Edition, Pearson Education (2009)
5. Hu, P.H., Lu, Z.X., Zhang, Y.Q. Liu, S.L., Dang, X.M.: A New Modeling Method of Angle Measurement for Intelligent Ball Joint Based on BP-RBF Algorithm. Applied Sciences 9(14), 2850 (2019) https://doi.org/10.3390/app9142850
6. Li, Q., Xiong, Q., Ji, S., Yu, Y., Wu, C., Yi, H.: A method for mixed data classification base on RBF-ELM network. Neurocomputing 431, 7–22 (2021)
7. Markopoulos, A.P., Georgiopoulos, S., Manolakos, D.E.: On the use of back propagation and radial basis function neural networks in surface roughness prediction. J Ind Eng Int 12, 389–400 (2016) https://doi.org/10.1007/s40092-016-0146-x
8. Meisel, W.S.: Potential Functions in Mathematical Pattern Recognition. IEEE Trans. Computers 18(10): 911-918 (1969)
9. Pedregosa, F. et al.: Scikit-learn: Machine learning in Python. Journal of Machine Learning Research 12(Oct), 2825–2830 (2011).
10. Siouda, R., Nemissi, M., Seridi, H.: Diverse activation functions based-hybrid RBF-ELM neural network for medical classification. Evol. Intel. (2022) https://doi.org/10.1007/s12065-022-00758-3
11. Wen, H., Xie, W., Pei, J.: A Structure-Adaptive Hybrid RBF-BP Classifier with an Optimized Learning Strategy. PLoS ONE 11(10) (2016) e0164719. https://doi.org/10.1371/journal.pone.0164719
12. Wen, H., Xie, W., Pei, J.: An incremental learning algorithm for the hybrid RBF-BP network classifier. EURASIP J. Adv. Signal Process 57 (2016). https://doi.org/10.1186/s13634-016-0357-8
13. Wen, H., Fan, H., Xie, W. and Pei, J.: Hybrid Structure-Adaptive RBF-ELM Network Classifier, in IEEE Access, vol. 5, pp. 16539-16554, (2017) doi: 10.1109/ACCESS.2017.2740420.
14. Wen, H., Yan, T., Liu, Z., Chen, D.: Integrated neural network model with pre-RBF kernels. Science Progress 104(3) (2021) doi:10.1177/00368504211026111
15. Wen, H., Li, T., Chen, D., Yang, J., Che, Y.: An optimized neural network classification method based on kernel holistic learning and division. Mathematical problems in engineering (2021) https://doi.org/10.1155/2021/8857818