Article

Pattern Classification Based on RBF Networks with Self-Constructing Clustering and Hybrid Learning

Zan-Rong He ¹, Yan-Ting Lin ¹, Chen-Yu Wu ¹, Ying-Jie You ¹ and Shie-Jue Lee ^{2,*}

¹ Department of Electrical Engineering, National Sun Yat-Sen University, Kaohsiung 80424, Taiwan; zlhe@water.ee.nsysu.edu.tw (Z.-R.H.); ytlin@itlm.ee.nsysu.edu.tw (Y.-T.L.); cywu@itlm.ee.nsysu.edu.tw (C.-Y.W.); Inyu@itlm.ee.nsysu.edu.tw (Y.-J.Y.)

² Department of Electrical Engineering and Intelligent Electronic Commerce Research Center, National Sun Yat-Sen University, Kaohsiung 80424, Taiwan

* Correspondence: leesj@mail.ee.nsysu.edu.tw

Received: 22 July 2020; Accepted: 23 August 2020; Published: 25 August 2020

Abstract: Radial basis function (RBF) networks are widely adopted to solve problems in the field of pattern classification. However, in the construction phase of such networks, there are several issues encountered, such as the determination of the number of nodes in the hidden layer, the form and initialization of the basis functions, and the learning of the parameters involved in the networks. In this paper, we present a novel approach for constructing RBF networks for pattern classification problems. An iterative self-constructing clustering algorithm is used to produce a desired number of clusters from the training data. Accordingly, the number of nodes in the hidden layer is determined. Basis functions are then formed, and their centers and deviations are initialized to be the centers and deviations of the corresponding clusters. Then, the parameters of the network are refined with a hybrid learning strategy, involving hyperbolic tangent sigmoid functions, steepest descent backpropagation, and least squares method. As a result, optimized RBF networks are obtained. With this approach, the number of nodes in the hidden layer is determined and basis functions are derived automatically, and higher classification rates can be achieved. Furthermore, the approach is applicable to construct RBF networks for solving both single-label and multi-label pattern classification problems.

Keywords: pattern classification; RBF network; self-constructing clustering; basis function; steepest descent backpropagation; least squares method

1. Introduction

Pattern classification is a process related to categorization, concerned with classifying patterns into one or more categories. It plays an important role in many fields, such as media, medicine, science, business, and security [1–5]. A classification problem can be converted to multiple binary classification problems in two ways, one-vs.-rest (OvR) and one-vs.-one (OvO), by creating several systems each of which is responsible for one category. However, OvR may suffer from unbalanced learning, since typically the set of negative training data is much larger than the set of positive training data. In contrast, OvO may suffer from ambiguities: two or more categories may receive the same number of votes and one category has to be picked arbitrarily. Therefore, researchers tend to seek other ways to solve classification problems.

In general, classification tasks can be divided into two kinds: single-label and multi-label. Single-label classification concerns classifying a pattern to only one category, but multi-label classification may classify a pattern to one or more categories. Many algorithms, based on multilayered perceptrons (MLP) [6], decision trees [7,8], k-nearest neighbors (KNN) [9], probability-based

classifiers [10], support vector machines (SVM) [11,12], or random vector functional-link (RVFL) networks [13,14], have been developed for pattern classification.

Most developed algorithms are for single-label classification. KNN [9] is a lazy learning algorithm. It is based on similarity between input patterns and training data. The category to which the input pattern belongs to is determined by its k-nearest neighbors. Decision tree methods [7,15] build decision trees, e.g., ID3 and C4.5, from a set of training data using the concept of information entropy. An unseen pattern traverses down from the root of the tree until reaching a leaf which tells the category to which the unseen is assigned. Probability-based classifiers [16,17] assume independence among features. These classifiers can be trained for the estimation of parameters necessary for classification. MLP [18–20] is a feedforward multi-layer network model consisting of several layers of nodes, with two adjacent layers fully connected to each other. Each node in the network is associated with an activation function, linear or nonlinear. RVFL networks [13,14,21] are also fully-connected feedforward networks, with only one single hidden layer. The weights between input and hidden nodes are assigned to random values. The values of the weights between hidden and output nodes, instead, are learned from training patterns. An RBF network [22–27] is a two-layer network, using radial basis functions as the activation functions in the hidden layer. The output of the network is a linear combination of the outputs from the hidden layer. In SVM [11,12], training patterns are mapped into the points in a high-dimensional space and the points of different categories are separated in the space by a gap as wide as possible. For an unseen pattern, the same mapping is performed and the classification is predicted according to the side of the gap it falls on. For solving multi-label classification problems [28], two approaches are generally adopted [29]: In one approach, a multi-label classification task is transformed to several single-label classification tasks which are instead solved by single-label classification methods. In the other approach, the capability of a specific single-label classification algorithm is extended to handle directly the multi-label data [30–33].

Since first introduced in 1988 by Broomhead and Lowe [34], RBF networks have been becoming popular in classification applications. However, as pointed out in [34,35], in the construction phase of such networks, there are several issues encountered, such as the determination of the number of nodes in the hidden layer, the form and initialization of the basis functions, and the learning of the parameters involved in the networks. In this paper, we present a novel approach for constructing RBF networks for pattern classification problems. An iterative self-constructing clustering algorithm is used to produce a desired number of clusters from the training data. Accordingly, the number of nodes in the hidden layer is determined. Basis functions are then formed, and their centers and deviations are initialized to be the centers and deviations of the corresponding clusters. Then the parameters of the network are refined with a hybrid learning strategy, involving hyperbolic tangent sigmoid functions, steepest descent backpropagation and least squares method. As a result, optimized RBF networks are obtained. Our approach can offer advantages in practicality. The number of nodes in the hidden layer is determined and basis functions are derived automatically, and higher classification rates can be achieved through the hybrid learning process. Furthermore, the approach is applicable to construct RBF networks for solving both single-label and multi-label pattern classification problems. Experimental results have shown that the proposed approach can be used to solve classification tasks effectively.

We have been working on RBF networks for years, and have developed different techniques [26,27,36,37]. This paper is mainly based on the MS thesis of the first author, Z.-R. He (who preferred to use a different English name, Tsan-Jung He, in the thesis) [38], supervised by S.-J. Lee, with the following contributions:

1. An iterative self-constructing clustering algorithm is applied to determine the number of hidden nodes and associated basis functions.
2. The centers and deviations of the basis functions are refined through the steepest descent backpropagation method.
3. Tikhonov regularization is applied in the optimization process to maintain the robustness of the output parameters.

4. The hyperbolic tangent sigmoid function is used as the activation function of the output nodes when learning the parameters of basis functions.

The rest of this paper is organized as follows: Section 2 gives an overview of the related work. The proposed approach is described in detail in Sections 3–5. Basis functions are derived by clustering from training data and a hybrid learning is used to optimize network parameters. Experimental results are presented in Section 6. The effectiveness of the proposed approach on single-label and multi-label classification is demonstrated. Finally, concluding remarks are given in Section 7.

2. Related Work

Many single-label classification algorithms have been proposed. The extended nearest neighbor (ENN) method [39], unlike the classic KNN, makes the prediction in a “two-way communication” style. It considers not only what are the nearest neighbors of the input pattern, but also those of which the input pattern is their nearest neighbors. The Natural Neighborhood Based Classification Algorithm (NNBCA) [40] provides a good classification result without artificially selecting the neighborhood parameter. Unlike KNN, it predicts different k for different samples. An enhanced general fuzzy min-max neural network (EGFM) classification model is proposed in [19] to perform supervised classification of data. New hyperbox expansion, overlap, and contraction rules are used to overcome some unidentified cases in some regions. SaE-ELM [41] is an improved algorithm of RVFL networks. In SaE-ELM, the hidden node parameters are optimized by the self-adaptive differential evolution algorithm, whose trial vector generation strategies and associated control parameters are self-adapted in a strategy pool by learning from their previous experiences in generating promising solutions, and the network output weights are calculated using the Moore–Penrose generalized inverse. In [42], basis functions in an RBF network are interpreted as probability density functions. The weights are seen as prior probabilities. Models that output class conditional densities or mixture densities were proposed. In [24], an RBF network based on KNN with adaptive selection radius is proposed. An adaptive selection radius is calculated according to the population density sampling method. The RBF network is trained to locate the sound source by solving nonlinear equations about the time difference between arrival and source of the sound.

Several problem transformation methods exist for multi-label classification. One popular transformation method, called binary relevance [43], transforms the original training dataset into p datasets, where p is the number of categories involved with the dataset. Each resulting dataset contains all the patterns of the original dataset, with each pattern assigned to one of the two labels: “belonging to” or “not belonging to” a particular category. Since the resulting datasets are all single-labeled, any single-label classification technique is applicable to them. The label powerset (LP) transformation [44] method creates one binary classifier for every label combination present in the training set. For example, eight possible label combinations are created if the number of categories associated with the original dataset is three. Ensemble methods were developed to create multi-label ensemble classifiers [45,46]. A set of multi-class single-label classifiers are created. For an input example, each classifier outputs a single class. These predictions are then combined by an ensemble method.

Some classification algorithms/models have been adapted to the multi-label task, without requiring problem transformations. Clare [29] is an adapted C4.5 version for multi-label classification. Other decision tree classification methods based on multi-valued attributes were also developed [47]. Zhang et al. [32] propose several multi-label learning algorithms including back-propagation multi-label learning (BP-MLL) [48], multi-label k -nearest neighbors (ML-KNN) [49] and multi-label RBF (ML-RBF) [50]. BP-MLL is a multi-label version of the back-propagation neural network. To take care of multi-labels, label co-occurrence is incorporated into the pairwise ranking loss function. However, it has a complex global error function to be minimized. ML-KNN is a lazy learning algorithm which requires a big run-time search. ML-RBF is a multi-label RBF neural network which is an extension of the traditional RBF learning algorithm. Multi-label with Fuzzy Relevance Clustering (ML-FRC), proposed by Lee and Jiang [51], is a fuzzy relevance clustering based method for the task of multi-label

text categorization. Kurata et al. [52] treat some of the nodes in the final hidden layer as dedicated neurons for each training pattern assigned to more than one label. These dedicated neurons are initialized to connect to the corresponding co-occurring labels with stronger weights than to others. A multi-label metamorphic relations prediction approach, named RBF-MLMR [33], is proposed to find a set of appropriate metamorphic relations for metamorphic testing. It uses soot analysis tool to generate the control flow graph and labels. The extracted nodes and the path properties constitute multi-label data sets for the control flow graph. Then, a multi-label RBF network prediction model is established to predict for multiple metamorphic relations.

3. Proposed Approach

Let $X = \{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}) | 1 \leq i \leq N\}$ be a finite set of N training patterns, where $\mathbf{x}^{(i)} \in \mathbb{R}^n$ is an input vector with n feature values, i.e., $\mathbf{x}^{(i)} = [x_1^{(i)} x_2^{(i)} \cdots x_n^{(i)}]^T$, and $\mathbf{y}^{(i)}$ is the corresponding category vector with m components, i.e., $\mathbf{y}^{(i)} = [y_1^{(i)} y_2^{(i)} \cdots y_m^{(i)}]^T$, of pattern i , defined as

$$y_f^{(i)} = \begin{cases} +1, & \text{if } \mathbf{x}^{(i)} \text{ belong to category } f; \\ -1, & \text{if } \mathbf{x}^{(i)} \text{ does not belong to category } f \end{cases} \quad (1)$$

for $1 \leq f \leq m$. Note that m denotes the number of categories associated with the training dataset X . For convenience, the categories are labelled as $1, 2, \dots, m$ respectively. For single-label classification, one of the elements in $\mathbf{y}^{(i)}$ is $+1$ and the other $m-1$ elements are -1 . For a multi-label classification, two or more components in $\mathbf{y}^{(i)}$ can be $+1$. Pattern classification is concerned with constructing a predicting model from X , and for any input vector $\mathbf{x}$ the model can respond with output $\hat{\mathbf{y}}$ showing which categories $\mathbf{x}$ belongs to.

In this work, we construct RBF networks as predicting models for classification. The data for training and testing the models will be collected from the openly accessible websites. The number of hidden nodes will be determined by clustering on the training data. The parameters of an RBF network will be learned by applying a hybrid learning algorithm. The performance of the constructed models will be analyzed by five-fold cross validation. Experiments for comparing performance with other methods will be conducted.

Our RBF network architecture, shown in Figure 1, is a two-layer network consisting of one hidden layer and one output layer. There are n input nodes, each node receiving and feeding forward one feature value of the input vector to the hidden layer. The hidden layer has J nodes, each with a basis function as its activation function. Each input node is fully connected to every node in the hidden layer. The output layer has m nodes, each corresponding to one distinct category. The hidden and output layers are also fully connected, i.e., each node in the hidden layer is connected to each node in the output layer. The weight between node j , $1 \leq j \leq J$, of the hidden layer and node f , $1 \leq f \leq m$, of the output layer is denoted by w_{fj} for $1 \leq j \leq J$ and $1 \leq f \leq m$.

Figure 1. Architecture of the proposed network model.

When an input vector $\mathbf{x} = [x_1 x_2 \cdots x_n]^T$ is presented to the input nodes of the network, the network produces the output vector $\hat{\mathbf{y}} = [\hat{y}_1 \hat{y}_2 \cdots \hat{y}_m]^T$ at the output layer. The network operates as follows.

- Input node i , $1 \leq i \leq n$, takes x_i and dispatches it to every node in the hidden layer.
- Hidden layer. Hidden node j , $1 \leq j \leq J$, is responsible for computing the value of its designated basis function as

$$B_j(\mathbf{x}) = \prod_{k=1}^n \exp \left[- \left(\frac{x_k - c_{k,j}}{\alpha v_{k,j}} \right)^2 \right] \quad (2)$$

where $c_{k,j}$ and $v_{k,j}$ denote the center and deviation, respectively, of the k th dimension. Note that α is a constant for controlling the scope of the function. Let

$$h_j = B_j(\mathbf{x}) \quad (3)$$

which is forwarded to the output layer.

- Output layer. Node f , $1 \leq f \leq m$, in this layer calculates the weighted sum

$$s_f = w_{f,1}h_1 + w_{f,2}h_2 + \cdots + w_{f,J}h_J + b_f \quad (4)$$

where $w_{f,1}, w_{f,2}, \dots, w_{f,J}$ are the weights and b_f is the bias associated with it. Then it provides the network output $\hat{y}_f$ by

$$\hat{y}_f = \text{sign}(s_f) \quad (5)$$

where $\text{sign}(\cdot)$ is the sign function defined as

$$\text{sign}(s_f) = \begin{cases} +1, & \text{if } s_f \geq 0; \\ -1, & \text{if } s_f < 0. \end{cases} \quad (6)$$

Therefore, the network output $\hat{\mathbf{y}}$ for the input vector $\mathbf{x}$ is

$$\hat{\mathbf{y}} = \begin{bmatrix} \hat{y}_1 & \hat{y}_2 & \cdots & \hat{y}_m \end{bmatrix}^T \quad (7)$$

Each component in $\hat{\mathbf{y}}$ is either +1 or -1. If component f is +1, $1 \leq f \leq m$, $\mathbf{x}$ is predicted to belong to category f . Note that one component or more in $\hat{\mathbf{y}}$ can be +1. Therefore, the proposed network architecture is applicable to solving both single-label and multi-label pattern classification problems.

We describe below how the network of Figure 1 is created from the given training set. Two phases, network setup and parameter refinement, are involved. In the first phase, the network structure is built and the initial values of the parameters are set. Then the parameters of the network are optimally refined in the second phase.

4. Network Setup Phase

In this phase, the number of nodes and basis functions in the hidden layer are determined automatically. Furthermore, the centers and deviations of the basis functions, and the weights and biases associated with the output layer, are initialized appropriately.

4.1. Initialization of Basis Functions

To determine the number of nodes and basis functions in the hidden layer, we find a group of clusters in the training set X through a clustering process. Firstly, we divide the training set X into m training subsets $X_1, X_2, \dots, X_m$ as follows:

$$X_h = \left\{ \mathbf{x}^{(i)} \mid 1 \leq i \leq N \wedge y_h^{(i)} = 1 \right\} \quad (8)$$

for $1 \leq h \leq m$. Then, each subset is fed to an iterative self-constructing clustering algorithm [53] and is grouped into $K_1, \dots, K_m$ clusters, respectively. A cluster contains patterns which are similar to each other, and is characterized by its center and deviation.

Let $K = K_1 + \dots + K_m$. Therefore, we have K clusters in total and let them be denoted as $C_1, C_2, \dots, C_K$ with centers $\mathbf{c}_k = [c_{1,k} \dots c_{n,k}]^T$ and deviations $\mathbf{v}_k = [v_{1,k} \dots v_{n,k}]^T$, $1 \leq k \leq K$. Then we set the number of hidden nodes, J , to be equal to K , i.e., $J = K$. For hidden node j , its associated basis function, $B_j(\mathbf{x})$, is also determined, with its center and deviation initialized to be the center and deviation of cluster C_j , i.e., we set

$$B_j(\mathbf{x}) = \prod_{i=1}^n \exp \left[- \left(\frac{x_i - c_{i,j}}{\alpha v_{i,j}} \right)^2 \right] \quad (9)$$

for $1 \leq j \leq J$, with $c_{i,j}$ and $v_{i,j}$ being the center and deviation, respectively, in the i th dimension of C_j .

4.2. Initialization of Weights and Biases

Next, we derive initial values for the weights $w_{f,j}$, $1 \leq j \leq J$, $1 \leq f \leq m$, between the hidden layer and the output layer, and for the biases b_f , $1 \leq f \leq m$, in the output layer. Consider the sum of squared error of the whole training set X :

$$\sum_{i=1}^N \| \mathbf{y}^{(i)} - \hat{\mathbf{y}}^{(i)} \|^2 = \sum_{i=1}^N \sum_{f=1}^m \left[y_f^{(i)} - \hat{y}_f^{(i)} \right]^2 \quad (10)$$

By Equations (4) and (5), we have

$$\hat{y}_f^{(i)} = \text{sign} \left(s_f^{(i)} \right), \quad (11)$$

$$s_f^{(i)} = w_{1,f} h_1^{(i)} + w_{2,f} h_2^{(i)} + \dots + w_{J,f} h_J^{(i)} + b_f \quad (12)$$

where $h_j^{(i)} = B_j(\mathbf{x}^{(i)})$, for $1 \leq f \leq m$ and $1 \leq i \leq N$. For pattern i , we have

$$\begin{aligned} s_1^{(i)} &= w_{1,1} h_1^{(i)} + w_{1,2} h_2^{(i)} + \dots + w_{1,J} h_J^{(i)} + b_1 \\ s_2^{(i)} &= w_{2,1} h_1^{(i)} + w_{2,2} h_2^{(i)} + \dots + w_{2,J} h_J^{(i)} + b_2 \\ &\dots = \dots, \\ s_m^{(i)} &= w_{m,1} h_1^{(i)} + w_{m,2} h_2^{(i)} + \dots + w_{m,J} h_J^{(i)} + b_m \end{aligned} \quad (13)$$

In total, there are $N \times m$ such equations. Although only the sign of $s_f^{(i)}$ matters in Equation (11), to be robust we'd like $s_f^{(i)}$ as close to $y_f^{(i)}$, which is either +1 or -1, as possible. Therefore, we demand

$$\begin{aligned}
y_1^{(i)} &= w_{1,1}h_1^{(i)} + w_{1,2}h_2^{(i)} + \cdots + w_{1,J}h_J^{(i)} + b_1 \\
y_2^{(i)} &= w_{2,1}h_1^{(i)} + w_{2,2}h_2^{(i)} + \cdots + w_{2,J}h_J^{(i)} + b_2 \\
&\vdots \\
y_m^{(i)} &= w_{m,1}h_1^{(i)} + w_{m,2}h_2^{(i)} + \cdots + w_{m,J}h_J^{(i)} + b_m
\end{aligned} \tag{14}$$

for $1 \leq i \leq N$. Equation (14) can be rewritten as

$$\|\mathbf{AW} - \mathbf{Y}\|^2 \tag{15}$$

where

$$\mathbf{A} = \begin{bmatrix} h_1^{(1)} & h_1^{(2)} & \cdots & h_1^{(N)} \\ h_2^{(1)} & h_2^{(2)} & \cdots & h_2^{(N)} \\ \vdots & \vdots & \ddots & \vdots \\ h_J^{(1)} & h_J^{(2)} & \cdots & h_J^{(N)} \\ 1 & 1 & \cdots & 1 \end{bmatrix}^T, \tag{16}$$

$$\mathbf{W} = \begin{bmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,J} & b_1 \\ w_{2,1} & w_{2,2} & \cdots & w_{2,J} & b_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ w_{m,1} & w_{m,2} & \cdots & w_{m,J} & b_m \end{bmatrix}^T, \tag{17}$$

$$\mathbf{Y} = \begin{bmatrix} y_1^{(1)} & y_1^{(2)} & \cdots & y_1^{(N)} \\ y_2^{(1)} & y_2^{(2)} & \cdots & y_2^{(N)} \\ \vdots & \vdots & \ddots & \vdots \\ y_m^{(1)} & y_m^{(2)} & \cdots & y_m^{(N)} \end{bmatrix}^T \tag{18}$$

with size of $N \times (J + 1)$, $(J + 1) \times m$, and $N \times m$, respectively.

To improve the robustness of the system [54], instead of minimizing Equation (15), we would like to minimize the penalized residual sum of squares:

$$\|\mathbf{AW} - \mathbf{Y}\|^2 + \lambda \|\mathbf{W}\|^2 \tag{19}$$

where λ is a pre-defined non-negative coefficient to balance the robustness against collinearity and the capability of fitting the desired outputs well. To find the solution to the minimization problem, we first rewrite Equation (19) as

$$\|\mathbf{AW} - \mathbf{Y}\|^2 + \lambda \|\mathbf{W}\|^2 = \|\mathbf{AW} - \mathbf{Y}\|^2 + \|\sqrt{\lambda}\mathbf{IW} - \mathbf{0}\|^2 = \left\| \begin{bmatrix} \mathbf{A} \\ \sqrt{\lambda}\mathbf{I} \end{bmatrix} \mathbf{W} - \begin{bmatrix} \mathbf{Y} \\ \mathbf{0} \end{bmatrix} \right\|^2$$

where $\mathbf{I}$ is the $(J + 1) \times (J + 1)$ identity matrix and $\mathbf{0}$ is the $(J + 1) \times m$ zero matrix. Let $\mathbf{A}' = \begin{bmatrix} \mathbf{A} \\ \sqrt{\lambda}\mathbf{I} \end{bmatrix}$ and $\mathbf{Y}' = \begin{bmatrix} \mathbf{Y} \\ \mathbf{0} \end{bmatrix}$. Then Equation (19) becomes

$$\|\mathbf{A}'\mathbf{W} - \mathbf{Y}'\|^2 \tag{20}$$

By the least squares method [55], the solution minimizing Equation (19) is

$$\begin{aligned}\hat{\mathbf{W}} &= \left\{ (\mathbf{A}')^T \mathbf{A}' \right\}^{-1} \mathbf{A}' \mathbf{Y}' \\ &= \left\{ \begin{bmatrix} \mathbf{A} \\ \sqrt{\lambda} \mathbf{I} \end{bmatrix}^T \begin{bmatrix} \mathbf{A} \\ \sqrt{\lambda} \mathbf{I} \end{bmatrix} \right\}^{-1} \begin{bmatrix} \mathbf{A} \\ \sqrt{\lambda} \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{Y} \\ \mathbf{0} \end{bmatrix} \\ &= \left\{ \begin{bmatrix} \mathbf{A}^T & \sqrt{\lambda} \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{A} \\ \sqrt{\lambda} \mathbf{I} \end{bmatrix} \right\}^{-1} \begin{bmatrix} \mathbf{A}^T & \sqrt{\lambda} \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{Y} \\ \mathbf{0} \end{bmatrix} \\ &= \left\{ \mathbf{A}^T \mathbf{A} + \lambda \mathbf{I} \right\}^{-1} \mathbf{A}^T \mathbf{Y}\end{aligned}\quad (21)$$

Note that the solution is unique even when $\mathbf{A}$ has dependent columns. This solution, $\hat{\mathbf{W}}$, provides initial values for the weights $w_{f,j}$, $1 \leq j \leq J$, $1 \leq f \leq m$, and biases b_f , $1 \leq f \leq m$, which can be refined later.

5. Parameter Refinement Phase

In this phase, the optimal values of the parameters of the network are learned from the training dataset X . Such parameters include the centers $c_j = [c_{1,j} \cdots c_{n,j}]^T$ and deviations $v_j = [v_{1,j} \cdots v_{n,j}]^T$, $1 \leq j \leq J$, associated with the basis functions in the hidden layer, and the weights $w_{f,j}$, $1 \leq j \leq J$, $1 \leq f \leq m$, between the hidden layer and the output layer, and the biases and biases b_f , $1 \leq f \leq m$, associated with the output nodes in the output layer. A hybrid learning process is applied iteratively. In each iteration, we first treat all the weights and biases as fixed and use the steepest descent backpropagation to update the centers and deviations associated with the basis functions. Then we treat all the centers and deviations associated with the basis functions as fixed and use the least squares method to update the weights and biases associated with the output layer. The process is iterated until convergence is reached.

5.1. Centers and Deviations

First, we describe how to refine the centers and deviations associated with the basis functions in the hidden layer. To make the refinement possible, we express $\hat{y}_f$ in Equation (5) in terms of the hyperbolic tangent sigmoid function which is analytic, i.e.,

$$\hat{y}_f \approx a_f = g(s_f) = \frac{e^{\beta s_f} - e^{-\beta s_f}}{e^{\beta s_f} + e^{-\beta s_f}} \quad (22)$$

for $1 \leq f \leq m$, where β is a slope-controlling constant and s_f is defined in Equation (4). Note that a_f provides a good approximation to the sign function and it is differentiable at all points.

Let us denote the initial values of the centers and deviations, shown in Equation (9), as $c_{k,j}(0)$ and $v_{k,j}(0)$, $1 \leq k \leq n$, $1 \leq j \leq J$, and the initial values of the weights and biases, obtained in Equation (21), as $w_{f,i}(0)$ and $b_f(0)$, $1 \leq f \leq m$, $1 \leq i \leq J$. As in [56], the mean squared error for the training set X can be estimated as

$$\begin{aligned}\hat{E}(k) &= \|\mathbf{y}(k) - \hat{\mathbf{y}}(k)\|^2 \\ &\approx \|\mathbf{y}(k) - \mathbf{a}(k)\|^2 = \sum_{l=1}^m [y_l(k) - a_l(k)]^2\end{aligned}\quad (23)$$

where $(x(k), y(k))$ can be any pattern in the training dataset X . By steepest descent backpropagation, the centers and deviations of the basis functions in the hidden layer are updated as

$$c_{i,j}(k+1) = c_{i,j}(k) - \eta \frac{\partial \hat{E}(k)}{\partial c_{i,j}}, 1 \leq i \leq n, 1 \leq j \leq J \quad (24)$$

$$v_{i,j}(k+1) = v_{i,j}(k) - \eta \frac{\partial \hat{E}(k)}{\partial v_{i,j}}, 1 \leq i \leq n, 1 \leq j \leq J \quad (25)$$

for $k \geq 0$, where η is the learning rate. From Equation (23), we have

$$\frac{\partial \hat{E}(k)}{\partial c_{i,j}} = -2 \sum_{l=1}^m [y_l(k) - a_l(k)] \frac{\partial a_l(k)}{\partial c_{i,j}} \quad (26)$$

$$\frac{\partial \hat{E}(k)}{\partial v_{i,j}} = -2 \sum_{l=1}^m [y_l(k) - a_l(k)] \frac{\partial a_l(k)}{\partial v_{i,j}} \quad (27)$$

By chain rule, we have

$$\frac{\partial a_l(k)}{\partial c_{i,j}} = \frac{\partial a_l(k)}{\partial s_l} \frac{\partial s_l(k)}{\partial h_j} \frac{\partial h_j(k)}{\partial c_{i,j}} \quad (28)$$

Since $a_l(k) = g(s_l(k))$, $s_l(k) = w_{l,1}(k)h_1(k) + \dots + w_{l,J}(k)h_J(k) + b_l(k)$, and $h_j(k) = \prod_{r=1}^n e^{-\left(\frac{x_r(k) - c_{r,j}(k)}{\alpha v_{r,j}(k)}\right)^2}$, we have

$$\begin{aligned} \frac{\partial a_l(k)}{\partial s_l} &= \dot{g}(s_l(k)) \\ &= \frac{\partial [e^{\beta s_l(k)} - e^{-\beta s_l(k)}]}{\partial s_l} \frac{1}{e^{\beta s_l(k)} - e^{-\beta s_l(k)}} + [e^{\beta s_l(k)} - e^{-\beta s_l(k)}] \frac{\partial [e^{\beta s_l(k)} - e^{-\beta s_l(k)}]^{-1}}{\partial s_l} \\ &= \beta(1 - a_l^2(k)) \end{aligned} \quad (29)$$

$$\frac{\partial s_l(k)}{\partial h_j} = w_{l,j}(k) \quad (30)$$

$$\begin{aligned} \frac{\partial h_j(k)}{\partial c_{i,j}} &= \frac{\partial e^{-\left(\frac{x_i(k) - c_{i,j}(k)}{\alpha v_{i,j}(k)}\right)^2}}{\partial c_{i,j}} \prod_{r \neq i} e^{-\left(\frac{x_r(k) - c_{r,j}(k)}{\alpha v_{r,j}(k)}\right)^2} \\ &= \frac{\partial \left[-\left(\frac{x_i(k) - c_{i,j}(k)}{\alpha v_{i,j}(k)}\right)^2 \right]}{\partial c_{i,j}} e^{-\left(\frac{x_i(k) - c_{i,j}(k)}{\alpha v_{i,j}(k)}\right)^2} \prod_{r \neq i} e^{-\left(\frac{x_r(k) - c_{r,j}(k)}{\alpha v_{r,j}(k)}\right)^2} \\ &= \frac{2[x_i(k) - c_{i,j}(k)]}{[\alpha v_{i,j}(k)]^2} h_j(k) \end{aligned} \quad (31)$$

Similarly, we have

$$\frac{\partial a_l(k)}{\partial v_{i,j}} = \frac{\partial a_l(k)}{\partial s_l} \frac{\partial s_l(k)}{\partial h_j} \frac{\partial h_j(k)}{\partial v_{i,j}} \quad (32)$$

$$\frac{\partial h_j(k)}{\partial v_{i,j}} = 2 \frac{[x_i(k) - c_{i,j}(k)]^2}{\alpha^2 v_{i,j}^3(k)} h_j(k) \quad (33)$$

Therefore, we have

$$c_{i,j}(k+1) = c_{i,j}(k) + 4\eta \frac{x_i(k) - c_{i,j}(k)}{[\alpha v_{i,j}(k)]^2} h_j(k) \sum_{l=1}^m [y_l(k) - a_l(k)] \dot{g}(s_l(k)) w_{l,j}(k) \quad (34)$$

$$v_{i,j}(k+1) = v_{i,j}(k) + 4\eta \frac{[x_i(k) - c_{i,j}(k)]^2}{\alpha^2 v_{i,j}^3(k)} h_j(k) \sum_{l=1}^m [y_l(k) - a_l(k)] \dot{g}(s_l(k)) w_{l,j}(k) \quad (35)$$

for $1 \leq i \leq n, 1 \leq j \leq J$. In matrix form this becomes:

$$c_j(k+1) = c_j(k) + 4\eta h_j(k) \mathbf{D}_{1,j}(k) \mathbf{q}_{1,j}(k) \mathbf{w}_j(k)^T \dot{\mathbf{G}}(\mathbf{s}(k)) \mathbf{e}(k) \quad (36)$$

$$v_j(k+1) = v_j(k) + 4\eta h_j(k) \mathbf{D}_{2,j}(k) \mathbf{q}_{2,j}(k) \mathbf{w}_j(k)^T \dot{\mathbf{G}}(\mathbf{s}(k)) \mathbf{e}(k) \quad (37)$$

for $1 \leq j \leq J$, where

$$\mathbf{q}_{1,j}(k) = \begin{bmatrix} x_1(k) - c_{1,j}(k) & x_2(k) - c_{2,j}(k) & \cdots & x_n(k) - c_{n,j}(k) \end{bmatrix}^T \quad (38)$$

$$\mathbf{q}_{2,j}(k) = \begin{bmatrix} [x_1(k) - c_{1,j}(k)]^2 & [x_2(k) - c_{2,j}(k)]^2 & \cdots & [x_n(k) - c_{n,j}(k)]^2 \end{bmatrix}^T \quad (39)$$

$$\mathbf{D}_{1,j}(k) = \begin{bmatrix} \frac{1}{[\alpha v_{1,j}(k)]^2} & 0 & 0 & \cdots & 0 & 0 \\ 0 & \frac{1}{[\alpha v_{2,j}(k)]^2} & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & \frac{1}{[\alpha v_{n,j}(k)]^2} \end{bmatrix} \quad (40)$$

$$\mathbf{D}_{2,j}(k) = \begin{bmatrix} \frac{1}{[\alpha^2 v_{1,j}^3(k)]^2} & 0 & 0 & \cdots & 0 & 0 \\ 0 & \frac{1}{[\alpha^2 v_{2,j}^3(k)]^2} & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & \frac{1}{[\alpha^2 v_{n,j}^3(k)]^2} \end{bmatrix} \quad (41)$$

$$\mathbf{w}_j(k) = \begin{bmatrix} w_{1,j}(k) & w_{2,j}(k) & \cdots & w_{m,j}(k) \end{bmatrix}^T \quad (42)$$

$$\dot{\mathbf{G}}(\mathbf{s}(k)) = \begin{bmatrix} \dot{g}(s_1(k)) & 0 & 0 & \cdots & 0 & 0 \\ 0 & \dot{g}(s_2(k)) & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & \dot{g}(s_m(k)) \end{bmatrix} \quad (43)$$

$$\mathbf{e}(k) = \begin{bmatrix} y_1(k) - a_1(k) & y_2(k) - a_2(k) & \cdots & y_m(k) - a_m(k) \end{bmatrix}^T \quad (44)$$

Note that the algorithm described above is the stochastic steepest descent backpropagation which involves incremental training. However, we perform batch training in which the complete gradient is computed, after all inputs are applied to the network, before the centers and deviations are updated. To implement the batch training, in each iteration the individual gradients for all the inputs in the training set are averaged to get the total gradient, and the update equations become

$$c_{i,j}(k+1) = c_{i,j}(k) - \frac{\eta}{N} \sum_{p=1}^N \frac{\partial \hat{E}_p(k)}{\partial c_{i,j}} \quad (45)$$

$$v_{i,j}(k+1) = v_{i,j}(k) - \frac{\eta}{N} \sum_{p=1}^N \frac{\partial \hat{E}_p(k)}{\partial v_{i,j}} \quad (46)$$

for $1 \leq i \leq n, 1 \leq j \leq J$, where $\hat{E}_p(k)$ is the mean squared error $\hat{E}(k)$ induced from pattern p , $(\mathbf{x}^{(p)}, \mathbf{y}^{(p)})$. In matrix form, the batch training becomes:

$$\mathbf{c}_j(k+1) = \mathbf{c}_j(k) + 4 \frac{\eta}{N} \sum_{p=1}^N \{h_j(p) \mathbf{D}_{1,j}(p) \mathbf{q}_{1,j}(p) \mathbf{w}_j(p)^T \dot{\mathbf{G}}(\mathbf{s}(p)) \mathbf{e}(p)\} \quad (47)$$

$$\mathbf{V}_j(k+1) = \mathbf{v}_j(k) + 4 \frac{\eta}{N} \sum_{p=1}^N \{h_j(p) \mathbf{D}_{2,j}(p) \mathbf{q}_{2,j}(p) \mathbf{w}_j(p)^T \dot{\mathbf{G}}(\mathbf{s}(p)) \mathbf{e}(p)\} \quad (48)$$

for $1 \leq j \leq J$.

5.2. Weights and Biases

Next, we describe how to refine weights and biases associated with the output layer in each iteration. For pattern i , $1 \leq i \leq N$, by Equation (13), we want

$$\begin{aligned} y_1^{(i)} &= w_{1,1}(k+1)h_1^{(i)}(k+1) + \dots + w_{1,J}(k+1)h_J^{(i)}(k+1) + b_1(b+1) \\ y_2^{(i)} &= w_{2,1}(k+1)h_1^{(i)}(k+1) + \dots + w_{2,J}(k+1)h_J^{(i)}(k+1) + b_2(b+1) \\ &\dots = \dots \\ y_m^{(i)} &= w_{m,1}(k+1)h_1^{(i)}(k+1) + \dots + w_{m,J}(k+1)h_J^{(i)}(k+1) + b_m(b+1) \end{aligned}$$

where

$$h_j^{(i)}(k+1) = \prod_{r=1}^n e^{-\left(\frac{x_r^{(i)} - c_{r,j}(k+1)}{av_{r,j}(k+1)}\right)^2} \quad (49)$$

for $1 \leq j \leq J$ and $1 \leq i \leq N$. Note that $c_{r,j}(k+1)$ and $v_{r,j}(k+1)$ are the newly updated values obtained in the first part of the current iteration. Let

$$\mathbf{A}(k+1) = \begin{bmatrix} h_1^{(1)}(k+1) & h_1^{(2)}(k+1) & \dots & h_1^{(N)}(k+1) \\ h_2^{(1)}(k+1) & h_2^{(2)}(k+1) & \dots & h_2^{(N)}(k+1) \\ \vdots & \vdots & \ddots & \vdots \\ h_J^{(1)}(k+1) & h_J^{(2)}(k+1) & \dots & h_J^{(N)}(k+1) \\ 1 & 1 & \dots & 1 \end{bmatrix}^T \quad (50)$$

and

$$\mathbf{W}(k+1) = \begin{bmatrix} w_{1,1}(k+1) & w_{1,2}(k+1) & \dots & w_{1,J}(k+1) & b_1(k+1) \\ w_{2,1}(k+1) & w_{2,2}(k+1) & \dots & w_{2,J}(k+1) & b_2(k+1) \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ w_{m,1}(k+1) & w_{m,2}(k+1) & \dots & w_{m,J}(k+1) & b_m(k+1) \end{bmatrix}^T \quad (51)$$

and $\mathbf{Y}$ be the same matrix as shown in Equation (18). As before, we'd like to minimize

$$\|\mathbf{A}(k+1)\mathbf{W}(k+1) - \mathbf{Y}\|^2 + \lambda \|\mathbf{W}(k+1)\|^2 \quad (52)$$

The solution is $\{\mathbf{A}^T(k+1)\mathbf{A}(k+1) + \lambda \mathbf{I}\}^{-1} \mathbf{A}^T(k+1)\mathbf{Y}$ which provides the new values of the weights and biases, i.e., $w_{f,j}(k+1)$ and $b_f(k+1)$, $1 \leq j \leq J$, $1 \leq f \leq m$.

6. Experimental Results

In this section, experimental results are presented to show the effectiveness of the proposed approach. Comparison results with other classification methods are also presented. For convenience, our approach is abbreviated as RBF-ISCC (RBF with Iterative Self-Constructing Clustering). We used a computer with Intel(R) Core(TM) i7 CPU@4.00GHz and 16 GB of RAM to conduct the experiments.

A five-fold cross validation was adopted to measure the performance of a classifier in the following experiments. Each dataset was randomly divided into five disjoint subsets, in such a way that the data of each category were evenly involved in five folds. Then, a classifier performed five runs for each dataset. In each run, four of the five subsets were used for training in the training phase and the remaining subset was used for testing in the testing phase. The results of the five runs were then averaged to indicate the performance of the classifier on the dataset. Note that the data for training were different from the data for testing in each run. The pre-defined constants, such as α of our method, were determined in the training phase and were not changed in the testing phase.

6.1. Experiment 1—Single-Label Data Sets

In this experiment, we would like to show the performance of single-label classification of different methods. Average accuracy (AACC), macro-averaged recall (MAREC), macro-averaged precision (MAPRE), and macro-averaged F-measure (MAFM) are adopted metrics for performance evaluation [57]. For category i , $1 \leq i \leq m$, let TP_i , FP_i , FN_i , TN_i are the number of true positives, false positives, false negatives, and true negatives, respectively. AACC, MAREC, MAPRE, and MAFM are defined as

$$\begin{aligned} \text{AACC} &= \frac{\sum_{i=1}^m \frac{TP_i + TN_i}{TP_i + FP_i + FN_i + TN_i}}{m} \\ \text{MAREC} &= \frac{\sum_{i=1}^m \frac{TP_i}{TP_i + FN_i}}{m} \\ \text{MAPRE} &= \frac{\sum_{i=1}^m \frac{TP_i}{TP_i + FP_i}}{m} \\ \text{MAFM} &= \frac{2 \times \text{MAREC} \times \text{MAPRE}}{\text{MAREC} + \text{MAPRE}} \end{aligned}$$

Clearly, higher values of the above metrics indicate a better classification. Several benchmark data sets are used in this experiment. Table 1 lists the characteristics of these data sets. For example, the dataset “Liver Disorders” contains 345 patterns, and each patterns has 7 features. There are two categories involved in the dataset. Note that cardinality denotes the average number of categories each pattern belongs to. Since each pattern belongs to one and only one category, the cardinality of the dataset is 1.000. The data sets of Table 1 are taken from the repository of the machine learning database at the University of California at Irvine (UCI) [58].

Table 1. Single-label data sets for experiments.

Data Set	# Features	# Categories	# Patterns	Cardinality
Liver Disorders	7	2	345	1.000
Heart	13	2	270	1.000
Ionosphere	34	2	351	1.000
Sonar	60	2	208	1.000
Iris	4	3	150	1.000
Lung Cancer	56	3	32	1.000
Soybean	35	4	47	1.000
Glass	9	6	214	1.000
Ecoli	8	8	336	1.000
Yeast-SL	8	10	1484	1.000

Tables 2 and 3 shows the testing AACC and MMFM, respectively, obtained by different methods, including LinearSVC [59], SVC-rbf (Support Vector Classifier with RBF kernel), SVM [11], NN (Neural Network) [60], RBF [22], and RBF-ISCC for the data sets in Table 1. SVC-rbf, support vector classification with RBF kernel, is implemented based on *libsvm* [61] and using one-vs.-rest scheme for classification. LinearSVC, Linear support vector classification, is similar to SVC with linear kernel, but implemented in terms of *liblinear* [59] rather than *libsvm*. We also used one-vs.-rest scheme

for classification for LinearSVC. SVM is a support vector machine based on the MATLAB toolbox (<http://www.mathworks.com>). NN is a three-layer neural network, using the hyperbolic tangent sigmoid transfer function (tanh) in the hidden layer. NN optimizes the log-loss function using LBFGS (Limited-memory BFGS) [60], which is a BFGS (Broyden–Fletcher–Goldfarb–Shanno) based optimizer in the family of quasi-Newton methods. The codes of LinearSVC, SVC-rbf, and NN are downloaded from the scikit-learn library (<http://scikit-learn.org>) [62,63]. RBF is a radial basis function network downloaded from the website <http://neupy.com/apidocs>. In the tables, the best performance for each data set is boldfaced. The average metric values obtained by each method are shown in the bottom line of Tables 2 and 3. Note that RBFISCC performs best, having the highest average AACC, 87.72% and the highest average MAFM, 77.96%. In particular, RBF-ISCC gets the highest AACC and MAFM for 8 and 5, respectively, out of 10 data sets. Tables 4 and 5 show the MAREC and MAPRE values, respectively, obtained by different methods for all the data sets.

For these data sets, we set $\eta = 0.1$ in RBF-ISCC. The other constants are shown in Table 6. As mentioned earlier, their values are determined in the training phase and are not changed in the testing phase.

Table 2. Testing average accuracy (AACC) obtained for single-label data sets.

Data Set	LinearSVC	SVC-RBF	Support Vector Machines (SVM)	NN	Radial Basis Function (RBF)	RBF with Iterative Self-Constructing Clustering (RBF-ISCC)
Liver Disorders	0.7246	0.6812	0.7101	0.6232	0.5507	0.7536
Heart	0.7593	0.7593	0.7778	0.7407	0.7037	0.7778
Ionosphere	0.8873	0.9014	0.9014	0.8873	0.8310	0.9296
Sonar	0.8095	0.8571	0.8810	0.8810	0.8095	0.8333
Iris	0.9111	0.9778	0.9778	0.9778	0.9778	0.9778
Lung Cancer	0.5714	0.6190	0.5238	0.7143	0.4286	0.7143
Soybean	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
Glass	0.8837	0.8915	0.7907	0.8915	0.8915	0.8992
Ecoli	0.9632	0.9596	0.9044	0.9485	0.9412	0.9614
Yeast-SL	0.9101	0.9232	0.8532	0.9051	0.8956	0.9249
Ave-AACC	0.8420	0.8570	0.8320	0.8569	0.8030	0.8772

Table 3. Testing macro-averaged F-measure (MAFM) obtained for single-label data sets.

Data Set	LinearSVC	SVC-rbf	SVM	NN	RBF	RBF-ISCC
Liver Disorders	0.7189	0.6662	0.6983	0.6134	0.5599	0.7461
Heart	0.7587	0.7587	0.7768	0.7373	0.6973	0.7776
Ionosphere	0.8768	0.8920	0.8920	0.8750	0.8119	0.9266
Sonar	0.8132	0.8623	0.8808	0.8808	0.8091	0.8337
Iris	0.8421	0.9682	0.9682	0.9682	0.9682	0.9682
Lung Cancer	0.4915	0.2000	0.1667	0.6377	0.1333	0.5000
Soybean	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
Glass	0.6266	0.6325	0.4332	0.6015	0.7213	0.6325
Ecoli	0.8936	0.8679	0.5893	0.7079	0.7281	0.8663
Yeast-SL	0.4251	0.6614	0.2427	0.5426	0.5111	0.5450
Ave-MAFM	0.7447	0.7509	0.6648	0.7564	0.6940	0.7796

Table 4. Testing macro-averaged recall (MAREC) obtained for single-label data sets.

Data Set	LinearSVC	SVC-rbf	SVM	NN	RBF	RBF-ISCC
Liver Disorders	0.7198	0.6586	0.6931	0.6134	0.5603	0.7448
Heart	0.7417	0.7417	0.7625	0.7250	0.6917	0.7792
Ionosphere	0.8491	0.8691	0.8691	0.8674	0.7783	0.9365
Sonar	0.8048	0.8523	0.8795	0.8795	0.8091	0.8341
Iris	0.8000	0.9667	0.9667	0.9667	0.9667	0.9667
Lung Cancer	0.6111	0.3333	0.3333	0.6111	0.1667	0.5000
Soybean	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
Glass	0.6063	0.5913	0.3786	0.5730	0.7071	0.5611
Ecoli	0.8548	0.8470	0.5881	0.7132	0.7000	0.8235
Yeast-SL	0.3593	0.6280	0.2326	0.5692	0.5524	0.4280
Ave-MAREC	0.7347	0.7488	0.6704	0.7519	0.6932	0.7574

Table 5. Testing macro-averaged precision (MAPRE) obtained for single-label data sets.

Data Set	LinearSVC	SVC-rbf	SVM	NN	RBF	RBF-ISCC
Liver Disorders	0.7179	0.6739	0.7036	0.6134	0.5594	0.7474
Heart	0.7766	0.7766	0.7917	0.7500	0.7029	0.7761
Ionosphere	0.9064	0.9162	0.9162	0.8827	0.8486	0.9169
Sonar	0.8221	0.8726	0.8822	0.8822	0.8091	0.8333
Iris	0.8889	0.9697	0.9697	0.9697	0.9697	0.9697
Lung Cancer	0.4111	0.1429	0.1111	0.6667	0.1111	0.5000
Soybean	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
Glass	0.6482	0.6799	0.5062	0.6333	0.7361	0.7250
Ecoli	0.9144	0.8898	0.5905	0.7027	0.7586	0.9139
Yeast-SL	0.5205	0.6989	0.2537	0.5183	0.4755	0.7499
Ave-MAPRE	0.7606	0.7621	0.6725	0.7619	0.6971	0.8132

Table 6. Pre-defined constants for RBF-ISCC for single-label data sets.

Data Set	α	β	λ		α	β	λ
Liver Disorders	4	2	0	Heart	4	8	0.1
Ionosphere	4	2	0	Sonar	4	8	0
Iris	4	4	0	Lung Cancer	6	2	0
Soybean	4	2	0	Glass	2	2	0.1
Ecoli	4	16	0	Yeast-SL	4	16	0

6.2. Experiment 2—Multi-Label Data Sets

In this experiment, we show the performance of different methods on multi-label classification datasets. Exact match ratio (EMR), labeling F-measure (LFM), and Hamming loss (HL) are adopted for performance evaluation [57]. Let N_t be the number of patterns. EMR, LFM, and HL are defined as

$$\begin{aligned}
 \text{EMR} &= \frac{\sum_{i=1}^{N_t} I(\mathbf{y}^{(i)}, \hat{\mathbf{y}}^{(i)})}{N_t} \\
 \text{LFM} &= \frac{\sum_{i=1}^{N_t} \frac{2 \sum_{h=1}^m \left(\frac{1+y_h^{(i)}}{2} \right) \left(\frac{1+\hat{y}_h^{(i)}}{2} \right)}{\sum_{h=1}^m \left[\frac{1+y_h^{(i)}}{2} + \frac{1+\hat{y}_h^{(i)}}{2} \right]}}{N_t} \\
 \text{HL} &= \frac{\sum_{i=1}^{N_t} H_d(\mathbf{y}^{(i)}, \hat{\mathbf{y}}^{(i)})}{mN_t}
 \end{aligned}$$

where $I(\mathbf{a}, \mathbf{b}) = 1$ if $\mathbf{a}$ is identical to $\mathbf{b}$ and $I(\mathbf{a}, \mathbf{b}) = 0$ otherwise, and $H_d(\mathbf{a}, \mathbf{b})$ is the Hamming distance between $\mathbf{a}$ and $\mathbf{b}$. Note that higher values in EMR and LFM, and lower value in HL, indicate a better classification. Several benchmark data sets, taken from the MULAN (Multi-Label Learning) library [30], are used in this experiment. The characteristics of these data sets are shown in Table 7. A pattern may belong to more than one category in average, so the cardinality of a data set can be greater than 1. For instance, the cardinality of “Yeast-ML” is 4.237, indicating each pattern belonging to 4.237 categories in average.

Table 7. Multi-label data sets for experiments.

Data Set	# Features	# Categories	# Patterns	Cardinality
Birds	260	19	645	1.014
Scene	294	6	2407	1.074
Emotions	72	6	593	1.869
Flags	19	7	194	3.392
Yeast-ML	103	14	2417	4.237

Tables 8–10 show the testing EMR, LFM, and HL obtained by different methods, including ML-KNN (K-nearest neighbors classifier for multi-label) [49], MLPC (Multi-layer Perceptron classifier) [64], ML-SVC (Support vector classifier for multi-label), ML-RBF (RBF network for multi-label) [50], and RBF-ISCC, for the data sets in Table 7. The codes of ML-KNN, MLPC, and ML-SVC are downloaded from the scikitlearn library, while the codes of ML-RBF is downloaded from the public domain: <http://cse.seu.edu.cn/people/zhangml/resources.htm>. For these data sets, we set $\eta = 0.1$ in RBF-ISCC, except $\eta = 1.2$ for “Scene”. The other constants are shown in Table 11. Note that their values are determined in the training phase and are not changed in the testing phase.

The average metric values obtained by each method are shown in the bottom line of Tables 8–10. From these tables, we can see that RBF-ISCC performs the best in average EMR and average HL for these data sets. Table 12 ranks different methods based on the average results in Tables 8–10. Clearly, RBF-ISCC is ranked the first place.

Table 8. Testing exact match ratio (EMR) obtained for multi-label data sets.

Data Set	ML-KNN	MLPC	ML-SVC	ML-RBF	RBF-ISCC
Emotions	0.1832	0.2673	0.2030	0.2129	0.3020
Scene	0.4576	0.5518	0.3721	0.5418	0.5360
Yeast-ML	0.1045	0.0788	0.1368	0.0305	0.1526
Birds	0.4675	0.4675	0.4180	0.4892	0.5232
Flags	0.1538	0.1538	0.1385	0.1385	0.1846
Ave-EMR	0.2733	0.3038	0.2537	0.2826	0.3400

Table 9. Testing labeling F-measure (LFM) obtained for multi-label data sets.

Data Set	ML-KNN	MLPC	ML-SVC	ML-RBF	RBF-ISCC
Emotions	0.4812	0.6266	0.5716	0.4926	0.5893
Scene	0.4880	0.6734	0.5545	0.5924	0.5860
Yeast-ML	0.5633	0.5412	0.6024	0.4066	0.6203
Birds	0.4675	0.6225	0.5737	0.5372	0.5890
Flags	0.6329	0.6885	0.6830	0.6955	0.6827
Ave-LFM	0.5270	0.630	0.5970	0.5449	0.6135

Table 10. Testing Hamming loss (HL) obtained for multi-label data sets.

Data Set	ML-KNN	MLPC	ML-SVC	ML-RBF	RBF-ISCC
Emotions	0.2302	0.2137	0.2343	0.2195	0.2021
Scene	0.1130	0.1142	0.1568	0.0984	0.0994
Yeast-ML	0.2136	0.2605	0.2108	0.2711	0.2017
Birds	0.0510	0.0595	0.1154	0.0838	0.0422
Flags	0.3099	0.2659	0.2791	0.2835	0.2967
Ave-HL	0.1835	0.1828	0.1993	0.1913	0.1684

Table 11. Pre-defined constants for RBF-ISCC for multi-label data sets.

Data Set	α	β	λ		α	β	λ
Emotions	7	2	0.1	Scene	12	4	0
Yeast-ML	7	4	0	Birds	10	2	0
Flags	2	2	10				

Table 12. Ranking of different methods for multi-label data sets.

Data Set	ML-KNN	MLPC	ML-SVC	ML-RBF	RBF-ISCC
Ave-EMR	4	2	5	3	1
Ave-LFM	5	1	3	4	2
Ave-HL	3	2	5	4	1
Ave-Rank	4.00	1.67	4.33	3.67	1.33

6.3. Discussions

Our proposed approach includes two phases, network setup and parameter refinement. In the network setup phase, the iterative self-constructing clustering algorithm is used to produce J clusters. Assume that P_1 iterations is performed. In each iteration, each n -dimensional instance is compared against each of the J existing clusters. Therefore, the complexity of this phase is $O(nJP_1)$. In the parameter refinement phase, a hybrid learning, integrating the steepest descent and least squares methods, is applied to refine the parameters associated with the network. Let P_2 be the number of iterations performed in this phase. The complexity of the least squares method is $O(NJ^2)$, while the complexity of steepest descent is $O(nNJ)$. Therefore, the complexity of the parameter refinement phase is $O(nJP_2 + NJ^2P_2)$. As a result, the total complexity of our approach is $O(nJP_1 + nJP_2 + NJ^2P_2)$. We feel that it may not be appropriate to compare the performance from the speed viewpoint among different methods. The programs of the other existing methods are taken from different websites. They might be implemented in different languages or with different optimizing mechanisms. For support-vector based classifiers, e.g., LinearSVC and SVC-rbf, the quadratic programming solver takes a complexity between $O(nN^2)$ and $O(nN^3)$ [11,59]. For NN with k hidden layers each having h neurons, the time complexity of backpropagation is $O(nNmh^kP)$ where P is the number of iterations [60]. For KNN, the brute-force implementation takes a complexity of $O(nN^2)$. The computational time of each method is usually of the order of seconds for data sets of small or moderate size.

High-dimensional datasets may cause data sparsity and computational intractability, leading to low forecasting accuracy of classification algorithms. Two strategies, correlation analysis and dimensionality reduction, can be employed to reduce the dimensionality of a dataset. By correlation analysis, the correlation, e.g., Pearson product-moment correlation coefficient [65], for each pair of variables, is calculated. Variables are sorted out according to their relevance to the target to be predicted. Four levels of correlation, uncorrelated, weakly correlated, moderately correlated, and strongly correlated, are defined [66]. It was shown in [67] that by ignoring weakly correlated or uncorrelated variables, forecasting accuracy can be improved. Principal component analysis (PCA) [68] is a popular method for dimensionality reduction. PCA reduces the dimensionality of a dataset while retaining the variation present in the dataset up to the maximum extent. This is done by transforming the original variables to a new set of variables known as the principal components. Dimensionality reduction is done by choosing a small number of principal components. Some strategies were proposed to determine an appropriate number of principal components. One common strategy chooses g principal components so that g is as small as possible and the cumulative energy is above a certain threshold.

To address the scalability concern about N , two methods—sampling [69] and clustering [70]—can be adopted. We may take a random sample, with size N_s , from the N instances, such that N_s is smaller than N . We may apply clustering to reduce the training set size. The N instances are grouped into clusters and the obtained clusters are used as the representatives for training the model. Another approach through exploiting the parallel or distributed facility of the computing system has long been adopted to address the scalability problem [71]. The N instances are divided and dispatched to different CPUs or computing nodes which can run simultaneously to improve the training efficiency of the classification task.

7. Concluding Remarks

We have presented a novel approach of constructing RBF networks for solving supervised classification problems. An iterative self-constructing clustering algorithm is used to determine the number of nodes in the hidden layer. Basis functions are formed, and their centers and deviations are initialized to be the centers and deviations of the corresponding clusters. Then, the parameters of the network are refined with a hybrid learning strategy, involving the steepest descent backpropagation and least squares method. Hyperbolic tangent sigmoid functions and Tikhonov regularization are employed. As a result, optimized RBF networks are obtained. The proposed approach is applicable to construct RBF networks for solving both single-label and multi-label pattern classification problems. Experimental results have shown that the proposed approach can be used to solve classification tasks effectively.

Note that all the dimensions of a training pattern are treated to be equally important, i.e., no different weights are involved. It might be beneficial to incorporate a kind of weighting mechanism which suggests different weights in different dimensions [72]. Furthermore, the clustering algorithm may encounter difficulties finding useful clusters in high-dimensional data sets. Dimensionality reduction [65,68–70] may need to be applied in this case. We will take these as our future work.

Author Contributions: Data curation, Z.-R.H., Y.-T.L., C.-Y.W., and Y.-J.Y.; Formal analysis, S.-J.L.; Funding acquisition, S.-J.L.; Methodology, Z.-R.H. and S.-J.L.; Project administration, S.-J.L.; Resources, Z.-R.H., Y.-T.L., C.-Y.W., and Y.-J.Y.; Software, Z.-R.H., Y.-T.L., C.-Y.W., and Y.-J.Y.; Validation, Z.-R.H. and S.-J.L.; Writing—original draft, Z.-R.H.; Writing—review & editing, S.-J.L. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by grants MOST-108-2221-E-110-046-MY2 and MOST-107-2622-E-110-008-CC3, Ministry of Science and Technology, and by the “Intelligent Electronic Commerce Research Center” from the Featured Areas Research Center Program within the framework of the Higher Education Sprout Project by the Ministry of Education (MOE) in Taiwan.

Acknowledgments: The anonymous reviewers are highly appreciated for their comments which were very helpful in improving the quality and presentation of the paper.

Conflicts of Interest: The authors declare no conflict of interest.

References

- Ozay, M.; Esnaola, I.; Vural, F.T.Y.; Kulkarni, S.R.; Poor, H.V. Machine learning methods for attack detection in the smart grid. *IEEE Trans. Neural Netw. Learn. Syst.* **2016**, *27*, 1773–1786. [\[CrossRef\]](#) [\[PubMed\]](#)
- Sutar, R.G.; Kothari, A.G. Intelligent electrocardiogram pattern classification and recognition using low-cost cardio-care system. *IET Sci. Meas. Technol.* **2015**, *9*, 134–143. [\[CrossRef\]](#)
- Leite, J.P.R.R.; Moreno, R.L. Heartbeat classification with low computational cost using Hjorth parameters. *IET Signal Process.* **2018**, *12*, 431–438. [\[CrossRef\]](#)
- Hajek, P.; Henriques, R. Mining corporate annual reports for intelligent detection of financial statement fraud—A comparative study of machine learning methods. *Knowl. Based Syst.* **2017**, *128*, 139–152. [\[CrossRef\]](#)
- Parisi, L.; RaviChandran, N.; Manaog, M.L. Decision support system to improve postoperative discharge: A novel multi-class classification approach. *Knowl. Based Syst.* **2018**, *152*, 1–10. [\[CrossRef\]](#)
- Haykin, S. *Neural Networks: A Comprehensive Foundation*, 1st ed.; Prentice Hall PTR: Upper Saddle River, NJ, USA, 1994.
- Quinlan, J.R. Simplifying decision trees. *Int. J. Hum. Comput. Stud.* **1999**, *51*, 497–510. [\[CrossRef\]](#)
- Ignatov, D.Y.; Ignatov, A.D. Decision stream: Cultivating deep decision trees. In Proceedings of the IEEE 29th International Conference on Tools for Artificial Intelligence, Boston, MA, USA, 6–8 November 2017; pp. 905–912.
- Aha, D.W. *Lazy Learning*; Springer: Dordrecht, The Netherlands, 1997.
- Russell, S.J.; Norvig, P. *Artificial Intelligence: A Modern Approach*; Pearson Education Limited: Kuala Lumpur, Malaysia, 2016.
- Cortes, C.; Vapnik, V. Support-vector networks. *Mach. Learn.* **1995**, *20*, 273–297. [\[CrossRef\]](#)
- Suykens, J.; Vandewalle, J. Least squares support vector machine classifiers. *Neural Process. Lett.* **1999**, *9*, 293–300. [\[CrossRef\]](#)

13. Schmidt, W.F.; Kraaijveld, M.A.; Duin, R.P.W. Feedforward neural networks with random weights. In Proceedings of the 11st IAPR International Conference on Pattern Recognition, Conference B: Pattern Recognition Methodology and Systems, Hague, The Netherlands, 30 August–3 September 1992; Volume II, pp. 1–4.
14. Pao, Y.; Park, G.H.; Sobajic, D.J. Learning and generalization characteristics of random vector functional-link net. *Neurocomputing* **1994**, *6*, 163–180. [[CrossRef](#)]
15. Karabadi, N.E.I.; Seridi, H.; Bousetouane, F.; Dhifli, W.; Aridhi, S. An evolutionary scheme for decision tree construction. *Knowl. Based Syst.* **2017**, *119*, 166–177. [[CrossRef](#)]
16. Liu, Z.G.; Pan, Q.; Mercier, G.; Dezert, J. A new incomplete pattern classification method based on evidential reasoning. *IEEE Trans. Cybern.* **2015**, *45*, 635–646. [[CrossRef](#)] [[PubMed](#)]
17. Dadaneh, S.Z.; Dougherty, E.R.; Qian, X. Optimal Bayesian classification with missing values. *IEEE Trans. Signal Process.* **2018**, *66*, 4182–4192. [[CrossRef](#)]
18. Liu, Z.; Zhuo, C.; Xu, X. Efficient segmentation method using quantised and non-linear CeNN for breast tumour classification. *Electron. Lett.* **2018**, *54*, 737–738. [[CrossRef](#)]
19. Donglikar, N.V.; Waghmare, J.M. An enhanced general fuzzy min-max neural network for classification. In Proceedings of the 2017 International Conference on Intelligent Computing and Control Systems (ICICCS), Melur, India, 15–16 June 2017; pp. 757–764.
20. Acharya, U.R.; Fujita, H.; Lih, O.S.; Adam, M.; Tan, J.H.; Chua, C.K. Automated detection of coronary artery disease using different durations of ECG segments with convolutional neural network. *Knowl. Based Syst.* **2017**, *132*, 62–71. [[CrossRef](#)]
21. Wang, C.-R.; Xu, R.-F.; Lee, S.-J.; Lee, C.-H. Network intrusion detection using equality constrained-optimization-based extreme learning machines. *Knowl. Based Syst.* **2018**, *147*, 68–80. [[CrossRef](#)]
22. Orr, M.J.L. Introduction to Radial Basis Function Networks. Ph.D. Thesis, University of Edinburgh, Edinburgh, UK, 1996.
23. Mak, M.-W.; Kung, S.-Y. Estimation of elliptical basis function parameters by the EM algorithm with application to speaker verification. *IEEE Trans. Neural Netw.* **2000**, *11*, 961–969.
24. Yang, X.; Li, Y.; Sun, Y.; Long, T.; Sarkar, T.K. Fast and robust RBF neural network based on global k-means clustering with adaptive selection radius for sound source angle estimation. *IEEE Trans. Antennas Propag.* **2018**, *66*, 3097–3107. [[CrossRef](#)]
25. Zhou, Q.; Wang, Y.; Jiang, P.; Shao, X.; Choi, S.-K.; Hu, J.; Cao, L.; Meng, X. An active learning radial basis function modeling method based on self-organization maps for simulation-based design problems. *Knowl. Based Syst.* **2017**, *131*, 10–27. [[CrossRef](#)]
26. Peng, H.-W.; Lee, S.-J.; Lee, C.-H. An oblique elliptical basis function network approach for supervised learning applications. *Appl. Soft Comput.* **2017**, *60*, 552–563. [[CrossRef](#)]
27. You, Y.-J.; Wu, C.-Y.; Lee, S.-J.; Liu, C.-K. Intelligent neural network schemes for multi-class classification. *Appl. Sci.* **2019**, *9*, 4036. [[CrossRef](#)]
28. Lapin, M.; Hein, M.; Schiele, B. Analysis and optimization of loss functions for multiclass, top-k, and multilabel classification. *IEEE Trans. Pattern Anal. Mach. Intell.* **2018**, *40*, 1533–1554. [[CrossRef](#)] [[PubMed](#)]
29. Madjarov, G.; Kocev, D.; Gjorgjevikj, D.; Džeroski, S. An extensive experimental comparison of methods for multi-label learning. *Pattern Recognit.* **2012**, *45*, 3084–3104. [[CrossRef](#)]
30. Tsoumakas, G.; Spyromitros-Xioufis, E.; Vilcek, J.; Vlahavas, I. MULAN: A Java library for multi-label learning. *J. Mach. Learn. Res.* **2011**, *12*, 2411–2414.
31. Lo, H.Y.; Lin, S.D.; Wang, H.M. Generalized k-labelsets ensemble for multi-label and cost-sensitive classification. *IEEE Trans. Knowl. Data Eng.* **2014**, *26*, 1679–1691.
32. Zhang, M.L.; Zhou, Z.H. A review on multi-label learning algorithms. *IEEE Trans. Knowl. Data Eng.* **2014**, *26*, 1819–1837. [[CrossRef](#)]
33. Zhang, P.; Zhou, X.; Pelliccione, P.; Leung, H. RBF-MLMR: A multi-label metamorphic relation prediction approach using RBF neural network. *IEEE Access* **2017**, *5*, 21791–21805. [[CrossRef](#)]
34. Broomhead, D.H.; Lowe, D. Multivariable functional interpolation and adaptive networks. *Complex Syst.* **1988**, *2*, 321–355.
35. Hagan, M.T.; Demuth, H.B.; Beale, M.H.; De Jesus, O. *Neural Network Design*, 2nd ed.; Martin Hagan: Notre Dame, IN, USA, 2012.
36. He, Z.-R.; Lin, Y.-T.; Lee, S.-J.; Wu, C.-H. A RBF network approach for function approximation. In Proceedings of the IEEE International Conference on Information and Automation, Wuyishan, China, 11–13 August 2018.

37. Tu, C.-S.; Wu, D.-Y.; Lee, S.-J.; Wu, C.-H. Regression estimation by radial basis function networks with self-constructing clustering. In Proceedings of the 5th International Conference on Systems and Informatics, Nanjing, China, 10–12 November 2018.
38. He, T.-J. A RBF Network Approach to Pattern Classification. Master's Thesis, National Sun Yat-Sen University, Kaohsiung, Taiwan, 2018.
39. Tang, B.; He, H. ENN: Extended nearest neighbor method for pattern recognition. *IEEE Comput. Intell. Mag.* **2015**, *10*, 52–60. [[CrossRef](#)]
40. Feng, J.; Wei, Y.; Zhu, Q. Natural neighborhood-based classification algorithm without parameter k. *Big Data Min. Anal.* **2018**, *1*, 257–265.
41. Wen, H.; Fan, H.; Xie, W.; Pei, J. Hybrid structure-adaptive RBF-ELM network classifier. *IEEE Access* **2017**, *5*, 16539–16554. [[CrossRef](#)]
42. Titsias, M.K.; Likas, A.C. Shared kernel models for class conditional density estimation. *IEEE Trans. Neural Netw.* **2001**, *12*, 987–997. [[CrossRef](#)] [[PubMed](#)]
43. Read, J.; Pfahringer, B.; Holmes, G.; Frank, E. Classifier chains for multi-label classification. *Mach. Learn.* **2011**, *85*, 333. [[CrossRef](#)]
44. Spolaor, N.; Cherman, E.A.; Monard, M.C.; Lee, H.D. A comparison of multi-label feature selection methods using the problem transformation approach. *Electron. Notes Theor. Comput. Sci.* **2013**, *292*, 135–151. [[CrossRef](#)]
45. Tsoumakas, G.; Vlahavas, I. Random k-labelsets: An ensemble method for multilabel classification. *Lect. Notes Artif. Intell.* **2007**, *4701*, 406–417.
46. Classification-Visualizers. Discrimination Threshold—Yellowbrick 0.9 Documentation. Available online: <http://www.scikit-yb.org> (accessed on 29 May 2020).
47. Li, H.; Guo, Y.J.; Wu, M.; Li, P.; Xiang, Y. Combine multi-valued attribute decomposition with multi-label learning. *Expert Syst. Appl.* **2010**, *37*, 8721–8728. [[CrossRef](#)]
48. Zhang, M.-L.; Zhou, Z.-H. Multilabel neural networks with applications to functional genomics and text categorization. *IEEE Trans. Knowl. Data Eng.* **2006**, *18*, 1338–1351. [[CrossRef](#)]
49. Zhang, M.-L.; Zhou, Z.-H. ML-KNN: A lazy learning approach to multi-label learning. *Pattern Recognit.* **2007**, *40*, 2038–2048. [[CrossRef](#)]
50. Zhang, M.-L. ML-RBF: RBF neural networks for multi-label learning. *Neural Process. Lett.* **2009**, *29*, 61–74. [[CrossRef](#)]
51. Lee, S.J.; Jiang, J.Y. Multilabel text categorization based on fuzzy relevance clustering. *IEEE Trans. Fuzzy Syst.* **2014**, *22*, 1457–1471. [[CrossRef](#)]
52. Kurata, G.; Xiang, B.; Zhou, B. Improved neural network-based multi-label classification with better initialization leveraging label co-occurrence. In Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego, CA, USA, 12–17 June 2016; pp. 521–526.
53. Wang, Z.-Y. Some Variants of Self-Constructing Clustering. Master's Thesis, National Sun Yat-Sen University, Kaohsiung, Taiwan, 2017.
54. Tikhonov, A.N.; Goncharsky, A.V.; Stepanov, V.V.; Yagola, A.G. *Numerical Methods for the Solution of Ill-Posed Problems*; Kluwer Academic Publishers: Amsterdam, The Netherlands, 1995.
55. Golub, G.H.; Van Loan, C.F. *Matrix Computations*; JHU Press: Baltimore, MD, USA, 2012; Volume 3.
56. Widrow, B.; Winter, R. Neural nets for adaptive filtering and adaptive pattern recognition. *IEEE Comput. Mag.* **1988**, *21*, 25–39. [[CrossRef](#)]
57. Sokolova, M.; Lapalme, G. A systematic analysis of performance measures for classification tasks. *Inf. Process. Manag.* **2009**, *45*, 427–437. [[CrossRef](#)]
58. Asuncion, A.; Newman, D. UCI Machine Learning Repository. 2007. Available online: <https://archive.ics.uci.edu/ml/> (accessed on 20 June 2018).
59. Fan, R.-E.; Chang, K.-W.; Hsieh, C.-J.; Wang, X.-R.; Lin, C.-J. LIBLINEAR: A library for large linear classification. *J. Mach. Learn. Res.* **2008**, *9*, 1871–1874.
60. Nawi, N.M.; Ransing, M.R.; Ransing, R.S. An improved learning algorithm based on the broyden-fletcher-goldfarb-shanno (BFGS) method for back propagation neural networks. In Proceedings of the 6th International Conference on Intelligent Systems Design and Applications, Jinan, China, 16–18 October 2006; Volume 1, pp. 152–157.
61. Chang, C.-C.; Lin, C.-J. LIBSVM: A library for support vector machines. *ACM Trans. Intell. Syst. Technol.* **2011**, *2*, 1–27. [[CrossRef](#)]

62. Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; et al. Scikitlearn: Machine learning in Python. *J. Mach. Learn. Res.* **2011**, *12*, 2825–2830.
63. Buitinck, L.; Louppe, G.; Blondel, M.; Pedregosa, F.; Mueller, A.; Grisel, O.; Niculae, V.; Prettenhofer, P.; Gramfort, A.; Grobler, J.; et al. API design for machine learning software: Experiences from the scikit-learn project. In Proceedings of the ECML PKDD Workshop: Languages for Data Mining and Machine Learning, Dublin, Ireland, 10–14 September 2013; pp. 108–122.
64. Glorot, X.; Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. In Proceedings of the 13th international conference on artificial intelligence and statistics, Sardinia, Italy, 13–15 May 2010; pp. 249–256.
65. Rodgers, J.L.; Nicewander, W.A. Thirteen ways to look at the correlation coefficient. *Am. Stat.* **1988**, *42*, 59–66. [[CrossRef](#)]
66. RMCAB. Bogotá Air Quality Monitoring Network, Website of Environmental Information. 2015. Available online: <http://201.245.192.252:81/> (accessed on 10 July 2020).
67. Li, L.; Wu, J.; Hudda, N.; Sioutas, C.; Fruin, S.A.; Delfino, R.J. Modeling the concentrations of on-road air pollutants in southern California. *Environ. Sci. Technol.* **2013**, *47*, 9291–9299. [[CrossRef](#)]
68. Wold, S.; Esbensen, K.; Geladi, P. Principal component analysis. *Chemom. Intell. Lab. Syst.* **1987**, *2*, 37–52. [[CrossRef](#)]
69. Wrobel, S. Scalability, search, and sampling: From smart algorithms to active discovery. In Proceedings of the European Conference on Principles of Data Mining and Knowledge Discovery, PKDD 2001, Freiburg, Germany, 3–5 September 2001; Lecture Notes in Computer Science 2168. De Raedt, L., Siebes, A., Eds.; Springer: Berlin/Heidelberg, Germany, 2001.
70. Liu, C.-F.; Yeh, C.-Y.; Lee, S.-J. A novel prototype reduction approach for supervised learning. *Int. J. Innov. Comput. Inf. Control* **2012**, *8*, 3963–3980.
71. Coulouris, G.; Dollimore, J.; Kindberg, T.; Blair, G. *Distributed Systems: Concepts and Design*, 5th ed.; Addison-Wesley: Boston, MA, USA, 2011.
72. Huang, X.; Ye, Y.; Xiong, L.; Lau, R.; Jiang, N.; Wang, S. Time series k-means: A new k-means type smooth subspace clustering for time series data. *Inf. Sci.* **2016**, *367*, 1–13. [[CrossRef](#)]

© 2020 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).