Neural Comput & Applic (2012) 21:305–317
DOI 10.1007/s00521-011-0546-1

ORIGINAL ARTICLE

Plane-Gaussian artificial neural network

Xubing Yang · Songcan Chen · Bin Chen

Received: 16 January 2010 / Accepted: 31 January 2011 / Published online: 22 February 2011

Abstract Multilayer perceptrons (MLPs) and radial basis functions networks (RBFNs) have been widely concerned in recent years. In this paper, based on *k*-plane clustering (*k*PC) algorithm, we propose a novel artificial network model termed as Plane-Gaussian network to enlarge the arsenal of the neural networks. This network adopts a so-called Plane-Gaussian activation function (PGF) in hidden neurons. Replacing traditional central point of Gaussian radial basis function (RBF) with central hyperplane, PGF forms a band-shaped rather than spheral-shaped receptive field in RBF, which makes PGF able to express its peculiar geometrical characteristics: locality and globality. Importantly, it is also proved that PGF network (PGFN) having one hidden layer is capable of universal approximation. As a universal approximator, PGFN gives an informal way of bridging the gap between MLP and RBFN. The experiments report comparison between training time and classification accuracies on some artificial and UCI datasets and conclude that (1) PGFN runs significantly faster than MLP and (2) PGFN has comparable or better classification performance than MLP and RBFN, especially in subspace-distributed datasets.

Keywords Multilayer perceptron (MLP) · Radial basis function (RBF) network · Activation function · Plane-Gaussian function (PGF)

X. Yang
College of Information Science and Technology,
Nanjing Forestry University, 210037 Nanjing,
People's Republic of China
e-mail: xbyang@nuaa.edu.cn

S. Chen (✉) · B. Chen
Department of Computer Science and Engineering,
Nanjing University of Aeronautics & Astronautics,
210016 Nanjing, People's Republic of China
e-mail: s.chen@nuaa.edu.cn

B. Chen
e-mail: b.chen@nuaa.edu.cn

B. Chen
Information Engineering College, Yangzhou University,
225009 Yangzhou, People's Republic of China

1 Introduction

MLPs and RBFNs, two types of classical multilayer feedforward neural networks, have been widely studied and got many successful applications such as in face recognition, disease diagnosis, risk investment, and so on [1–5], here just name a few. It had early been proved theoretically that they both are capable of approximating any continuous function or a mapping from an input space to an output space to arbitrary precision only if provided with sufficient hidden neurons [6–10]. However, in the real-world application, the number of samples is limited, thus in this case, any type of such networks with sufficiently large number of hidden nodes will lead to a serious overfitting. A remedy is to control the number. On the other hand, even such a number is selectively controlled, MLPs and RBFNs still result in different performances due to the adoption of their different activation functions [11, 12], which characterize different activating modes for the input space. In their concrete descriptions, MLP realizes a mapping by linearly combining a set of the *sigmoidal* functions that live on the hidden layer and have *global* activation domain, and while RBFN does likewise by linearly combining a set of the *exponential* functions with *local* activation domain (generally, Gaussian function), as illustrated, respectively, in Fig. 1a, b [13–17].

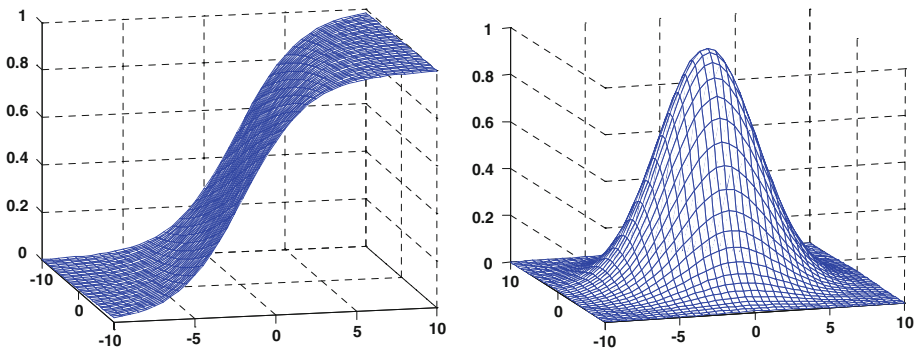

Fig. 1 3D visualization of MLP and RBFN's activation functions. **a** Logistic (sigmoidal) function, **b** Gaussian radial basis function

(a) Logistic (sigmoidal) function **(b)** Gaussian radial-basis function

In order to implement such networks given a set of limited training data, we can achieve this goal through a learning algorithm in terms of one of supervised, semi-supervised, and unsupervised-learning fashions. In this paper, we just concern the supervised learning but extend similar discussion to the other learnings. In such a scenario given limited training data, according to No Free Lunch (NFL) theorem [20], making sufficient use of prior information in data is one of effective ways to promote classification performance, here by prior information. We expect a so-constructed network having a good generalization, as have been analyzed for RBF networks. That is, in many given classification problems, the Gaussian activation functions are appropriate if having prior information that the distributions arise from a mixture of Gaussions. As for an architecture of a network, we will examine how the activating functions influence classification performance for data distributions or different structures (or mode) in data and then based on which develop a new type of activating function able to cater for such network. Since a hidden layer of activation function affords a distributed or global representation of the input, as foresaid, MLPs and RBFNs, respectively, take the sigmoidal functions and radial basis functions as their activating functions. Basically, MLP is typically trained in a supervised manner with a highly popular back-propagation (BP) algorithm and commonly takes the sigmoidal nonlinearity as its activation function to compute the local gradient in weight adjustment. Figure 1a seemingly shows that the sigmoidal function has global activating range and geometrically describes the globality of MLP without incorporating prior knowledge. Different from MLP, RBFN adopts the local activation function such as Gaussian function to accomplish such approximation. Concretely, the learning of RBFN usually depends on the following two steps [18]: the first is to compute the parameters i.e. the centers and widths of their activation functions (see Fig. 2).

For RBFN, *k*-means and FCM (fuzzy *c*-means) are the most typical methods to estimate cluster parameters in an unsupervised manner. Then, since the class label is available

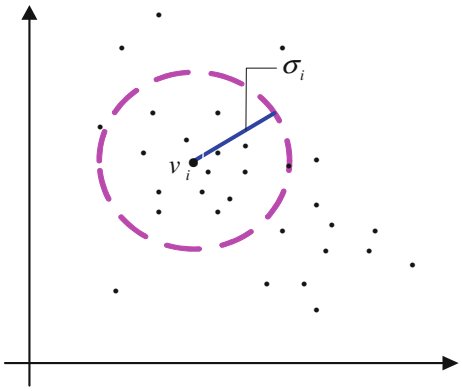

Fig. 2 Spheral-shaped field of Gaussian activation functions

in a given classification problem, many supervised methods could be used to construct foresaid sample-label mapping between the hidden and the output layers by the weighted linear combination of those Gaussian functions. Due to considering prior information, many researchers report that Gaussian RBF networks are appropriate if we have prior information that the distributions arise from a mixture of Gaussians (Fig. 3). That is, RBFNs achieve better overall performance than MLPs for the given spherical-/Gaussian-distributed data [19]. This is well consistent with NFL [20] which states that, for good generalization, there are no context-independent or usage-independent reasons to favor one classification method over another, unless appropriate prior information is incorporated in model selection.

But it has to be pointed out that for a real-world points from unknown distribution, how to select a suitable type of network to attain better generalization? Furthermore, the real-world data are complicated and may be from multifarious distributions, for instance, subspace distributions (two typical subspace-shaped distributions as illustrated in Fig. 4). To our knowledge, *k*-means and FCM are impressive for spherical distribution data, but powerless for such subspace-distributed data. Could we construct a network to meet such distribution? From the relationship between RBFN and *k*-means (or FCM), we know that there

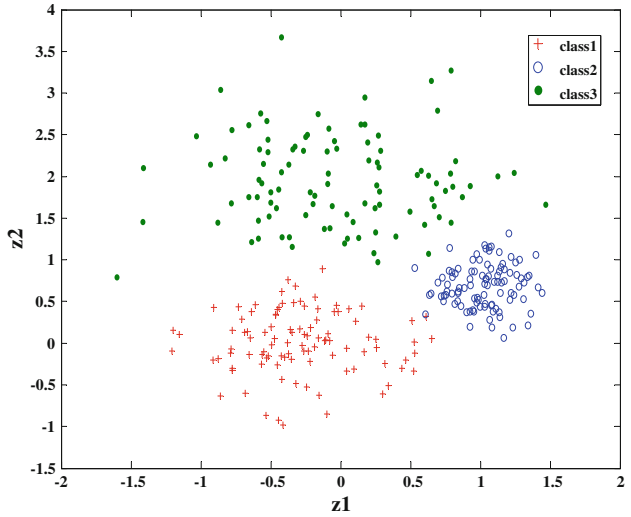

Fig. 3 Three-class data points sampled from Gaussian distributions (NormalData)

exist two major challenges. One is how to select an appropriate cluster algorithm to cater such distribution, and the other is network approximation, which is important for characterizing unknown-distributed data.

In this paper, we propose a network model under the guidance of NFL theorem. As foresaid, firstly an appropriate clustering algorithm must be selected to meet subspace distributions. Fortunately, there exactly exist two available subspace clustering algorithms such as FCV (Fuzzy c-Varieties) [21] and *k*PC (*k*-Plane Clustering) [22]. Both algorithms are effective for clustering subspace-distributed data though there exist differences between them in aspects such as initialization, type of prototypes, and computational complexity. In consideration of simplicity and efficiency, in this paper, we use *k*PC to directly generate corresponding hyperplane prototypes (similar to point prototypes in *k*-means). Through replacing the point prototypes (cluster centers) of Gaussian RBFN with those plane prototypes, we define a new activation function coined as Plane-Gaussian function (for short, PGF, see Fig. 5) and further develop a corresponding PGF network (PGFN). From Fig. 5, the activating range (belt-shaped field) of PGF presents two aspects: locality and globality. Geometrically, such locality is manifested in the limited bandwidth direction, and the globality is manifested in the unlimited spread along the orthogonal width direction.

The rest of this paper is organized as follows: Sect. 2 briefly introduces *k*PC algorithm. In Sect. 3, we describe the details of PGFN including network architecture, universal approximation, and, especially, relationship between PGFN and other networks. Section 4 reports our experimental results. Finally, we conclude the whole paper in Sect. 5.

2 *k*PC algorithm

This section presents a brief introduction for *k*-plane clustering algorithm.

Given a set of n points $x_j\,(j = 1, 2, \ldots, n)$ in the d-dimensional real space $\mathbb{R}^d$ represented by the matrix $\boldsymbol{X} \in \mathbb{R}^{d\times n}$. The goal of *k*PC algorithm is to cluster n points into c clusters corresponding to their nearest prototypes (hyperplanes) $\boldsymbol{P} = \{p_1, p_2, \ldots, p_c\}$. Based on this intuition, the objective function of *k*PC algorithm can be mathematically described as a nonconvex minimization problem as follows:

$$\begin{aligned} &\min J_{k\mathrm{PC}} = \sum_{i=1}^{c} \sum_{\mathbf{x}_j \in c_i} \left| w_i^T x_j - \gamma_i \right|^2 \\ &\text{s.t.} ||w_i|| = 1, \quad i = 1, 2, \ldots, c \end{aligned} \tag{1}$$

where c_i is the ith cluster composed by those points closest to the ith plane p_i, $\boldsymbol{w}_i$ and γ_i are the unit normal vector and threshold of the p_i, respectively. Throughout this paper, the superscript "T" denotes the transpose and e denotes a vector of ones of appropriate dimension.

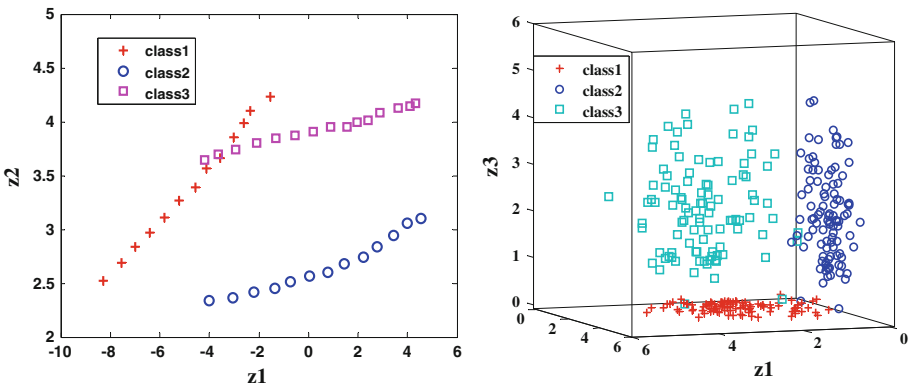

Fig. 4 Original subspace distributions. **a** Line subspace (Line3), **b** Plane subspce (Plane3)

(a) Line subspace (Line3) **(b)** Plane subspce (Plane3)

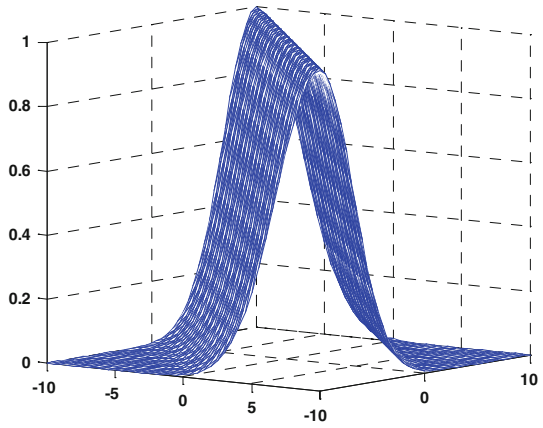

Fig. 5 The 3D visualization of the Plane-Gaussian function

By solving the above problem, *k*PC finds a set of hyperplanes $\{\boldsymbol{W}, \gamma\} = \{(\boldsymbol{w}_1, \gamma_1), (\boldsymbol{w}_2, \gamma_2), \ldots, (\boldsymbol{w}_c, \gamma_c)\}$. Thus, the hyperplanes $\boldsymbol{P} = \{p_1, p_2, \ldots, p_c\}$, i.e. c cluster prototypes can be determined. Then, each point can be assigned to its corresponding cluster set.

This algorithm alternates between assigning points to its nearest cluster hyperplane (shortly, Cluster Assignment) and for a given cluster, re-computing a cluster hyperplane under the objective function (1) (Cluster Update). Table 1 describes the *k*PC algorithm as below.

Next, based on the *k*PC algorithm, we come to construct so-called PGF network.

3 Plane-Gaussian network

In this section, we describe PGF network in the following parts: definition of PGF, network architecture, universal approximation, and relations with other networks.

3.1 Definition of Plane-Gaussian function

First, let us review something about activation functions. Note that general properties of activation functions are nonlinearity, continuity, boundness, and smoothness, but in real applications, some additional properties (such as monotonicity) are added to further promote the network performance [23]. One class of functions that has all the above properties is the sigmoid such as logistic function, which is defined below

$$h(\boldsymbol{x}) = \frac{1}{1 + \exp(-\boldsymbol{a}^T\boldsymbol{x} - b)} \tag{2}$$

where $\boldsymbol{a}$ and b denote the weight and bias of the neuron $\boldsymbol{x}$, respectively. Once the initial structure of MLP is determined and one learning round finished, the response of hidden neuron can be analyzed by this activation function [24]. If the distributions arise from a mixture of Gaussians, an appropriate function for RBFN is the multivariate Gaussian activation functions defined as follows:

$$\Phi^G(\boldsymbol{x}, v_i) = \exp\left(-\frac{||\boldsymbol{x} - \boldsymbol{v}_i||^2}{2\sigma_i^2}\right) \tag{3}$$

where $\boldsymbol{v}_i$ is the center position of the Gaussian function, σ_i is the width of the ith center associated with the ith cluster.

Table 1 *k*-Plane Clustering algorithm (*k*PC)

Input: Randomly initialize $\{(\boldsymbol{w}_1^1, \gamma_1^1), (\boldsymbol{w}_2^1, \gamma_2^1), \cdots, (\boldsymbol{w}_c^1, \gamma_c^1)\}$ satisfying $\|\boldsymbol{w}_i\| = 1\ (i = 1, \cdots, c)$.
Set the number of clusters c, initial iteration h=1 and size of training set n

Repeat
// Cluster Assignment
For j=1,…, n
Assign $\boldsymbol{x}_j$ to the closet hyperplane p_l. Determine l such that: $|\boldsymbol{w}_l^T\boldsymbol{x}_j - \gamma_l| = \min_{i=1,\cdots c} |\boldsymbol{w}_i^T\boldsymbol{x}_j - \gamma_i|$.
End
// Cluster Update
For i=1,.., c
Update the normal vector $\boldsymbol{w}_i^{h+1}$ to be an unit eigenvector corresponding to the smallest eigenvalue of $\boldsymbol{A}_i^T(\boldsymbol{I} - \frac{\boldsymbol{e}\boldsymbol{e}^T}{m_i})\boldsymbol{A}_i$; where $\boldsymbol{A}_i$ is a $d \times m_i$ matrix with columns corresponding to all points assigned to the i^{th} cluster, and m_i is the number of points in the ith cluster.
Update the threshold with $\gamma_i^{h+1} = \frac{\boldsymbol{e}^T\boldsymbol{A}_i\boldsymbol{w}_i^{h+1}}{m_i}$;
End
h=h+1;
until there is a repeated overall assignment of points to cluster hyperplanes.

Output: $\boldsymbol{W}$ and $\boldsymbol{\gamma}$.

Ref. [18] says that the performance of RBFN highly depends on the selections of the centers and widths, and current methods (such as gradient descent [25] and singular value decomposition [26]) of tuning these parameters cannot guarantee good classification results.

In this section, combining (2) and (3), we define a differentiable nonlinear function as follows:

Definition 1 A function with the following form is called Plane-Gaussian function (PGF),

$$\Phi^{\mathrm{PG}}(\boldsymbol{x}) = \exp\left(-\frac{|\boldsymbol{w}^T\boldsymbol{x}-\gamma|^2}{2\sigma^2}\right) \tag{4}$$

where $(\boldsymbol{w},\gamma)$ denotes a hyperplane $\boldsymbol{w}^T\boldsymbol{x}-\gamma=0$ with an unit normal vector $\boldsymbol{w}$ and a threshold γ, and σ is a width corresponding to the plane $(\boldsymbol{w},\gamma)$.

Figure 5 illustrates the PGF, and $|\boldsymbol{w}^T\boldsymbol{x}-\gamma|$ is the distance of the point $\boldsymbol{x}$ to the plane $(\boldsymbol{w},\gamma)$. Assume $\boldsymbol{x}$ is an element of the ith cluster associated with the plane $(\boldsymbol{w}_i,\gamma_i)$, σ_i denotes a semi-bandwidth (see Fig. 6), then the function $\Phi^{\mathrm{PG}}(\boldsymbol{x}) = \exp\left(-\frac{|\boldsymbol{w}_i^T\boldsymbol{x}-\gamma_i|^2}{2\sigma_i^2}\right)$ can be taken as the activation function of our proposed PGF network.

Intuitively, the so-constructed function $\Phi^{\mathrm{PG}}(\boldsymbol{x})$ is similar to that of logistic function (the weighted sum of all inputs plus the bias). Meanwhile, both PGF and RBF adopt the Euclidean distance to measure similarity and then accordingly assign the given points into their corresponding clusters. In Sect. 3.4, we will detail more about geometrical interpretation and their relations to both MLP and RBFNs.

3.2 Network architecture and training method

Without loss of generality, we merely discuss three-layer network architecture (one hidden layer) as illustrated in Fig. 7. In this graph, the input layer contains d neurons, each of which receives a d-dimensional sample attribute value, respectively. The second layer is a hidden layer, composed of nonlinear units that are connected directly to all of the nodes in the input layer. For PGFN, the activation function of the ith individual hidden unit is a PGF function described as Φ_i^{PG}. The output layer consists of simple linear units that are fully connected to the hidden layer. Similar to RBF, the weights of PGFN linking the input and the hidden layer are set all ones. When parameters $\boldsymbol{w}_i,\gamma_i$ and σ_i ($i=1,2,\ldots,c$) of PGFs are determined at the training phase, the weight matrix $\boldsymbol{U}$ (Fig. 7) between the hidden and output layer can be computed. Thus, the internal relationship between the input and output can also be viewed as a input–output-mapping $f(\boldsymbol{x}):\mathbb{R}^d\rightarrow\mathbb{R}^m$,

$$y = f(\boldsymbol{x}) = \boldsymbol{U}^T\boldsymbol{\Phi} = \boldsymbol{U}^T(\Phi_1^{\mathrm{PG}}(\boldsymbol{x}),\Phi_2^{\mathrm{PG}}(\boldsymbol{x}),\ldots,\Phi_c^{\mathrm{PG}}(\boldsymbol{x}))^T \tag{5}$$

where

$$\Phi_i^{\mathrm{PG}}(\boldsymbol{x}) \equiv \Phi_i^{\mathrm{PG}}(\boldsymbol{x},\boldsymbol{w}_i,\gamma_i,\sigma_i) = \exp\left(-\frac{|\boldsymbol{w}_i^T\boldsymbol{x}-\gamma_i|^2}{2\sigma_i^2}\right), \quad i=1,2,\ldots,c \tag{6}$$

As mentioned before, the parameters $\boldsymbol{w}_i$ and γ_i in (6) can be obtained by the kPC algorithm. The next issue is how to determine the unknown weight matrix $\boldsymbol{U}$. From the RBFN, we know that the linking weight matrix between the hidden and the output layer can be computed by the pseudo-inverse method [1]. This trick is also useful for training PGFN. Concretely, we divide the training phase into three steps as below.

Step 1: compute the outputs of the hidden layer.

Once the parameters of PGFs are determined, the outputs of the hidden layer can be computed and described in matrix form $\tilde{\Phi} = (\boldsymbol{\Phi}_1,\boldsymbol{\Phi}_2,\ldots,\boldsymbol{\Phi}_n)$, where

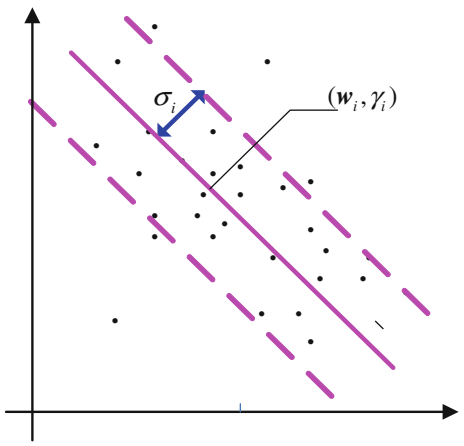

Fig. 6 Band-shaped field of Plane-Gaussian functions

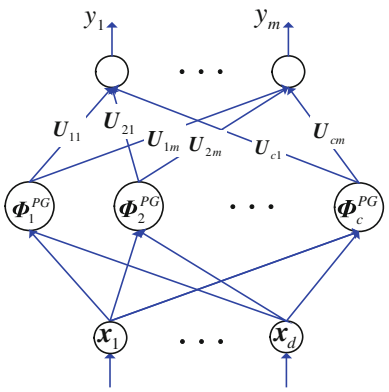

Fig. 7 Three-layer PGF neural network

$$\mathbf{\Phi}_j = (\Phi_1^{\mathrm{PG}}(\boldsymbol{x}_j),\ \Phi_2^{\mathrm{PG}}(\boldsymbol{x}_j), \ldots, \Phi_c^{\mathrm{PG}}(\boldsymbol{x}_j))^T,\ (j = 1, \ldots, n) \tag{7}$$

Step 2: set the target output matrix $\boldsymbol{Y}$ *using 0–1 encoding method.*

If a sample $\boldsymbol{x}_i$ belongs to the lth class, we define its corresponding 0–1 output code as a column vector $\boldsymbol{y}_i = (0, \ldots, 0, 1, 0, \ldots, 0)^T$, where only the lth component is set to 1. Accordingly, the 0–1 output matrix is termed as $\boldsymbol{Y} = (\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_n)$.

Step 3: calculate the weight matrix $\boldsymbol{U}$.

In this step, the foresaid pseudo-inverse trick is used to generate the linking weight matrix as

$$U = (\boldsymbol{Y}\widetilde{\Phi}^{+})^T \tag{8}$$

where $\boldsymbol{A}^+$ indicates the pseudo inverse of matrix $\boldsymbol{A}$. Generally, we set [1]

$$\boldsymbol{A}^+ = (\boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{A}^T \tag{9}$$

We rewrite foresaid training process in Table 2.

As a whole, training a PGFN is a process of determining the activation functions and the weight matrix $\boldsymbol{U}$. Thus, the aforementioned nonlinear input–output mapping can be constructed.

MLPs and RBFNs are typical examples of nonlinear layered feedforward networks. They are both universal approximators. That is, there always exists an RBF network capable of accurately mimicking a specified MLP or vice versa. Next, we theoretically analyze the approximation of our PGFN.

3.3 Universal approximation

From the theoretical point of view, the first thing we need to consider might be universal approximation theories. Weierstrass theorem [27] states function approximation on a *1*-dimensional (real-valued) case described as Lemma 1.

Table 2 Training algorithm of PGFN

Input: compute plane parameters with *k*PC algorithm; compute target output matrix with 0-1 code.
Training:
Step 1 determine the belt-width parameters σ_is by the Euclidean distances between the points of each cluster and their corresponding planes;
Step 2 compute output matrix of hidden layer $\tilde{\Phi}$ with (6) and (7);
Step 3 compute linking weight matrix $\boldsymbol{U}$ with (8) and (9)
Output: weighted matrix $\boldsymbol{U}$

Lemma 1 *If* f *is a continuous real-valued function on a closed interval [a, b], and if any* $\varepsilon > 0$ *is given, then there exists a polynomial* p *on [a, b] such that*

$$|f(x) - p(x)| < \varepsilon \tag{10}$$

for all $x \in [a, b]$.

Lemma 1 states that any continuous function over a closed interval on the real axis can be expressed in that interval as an absolutely and uniformly convergent series of polynomials to any degree of accuracy. Universal approximation theorem is based on Lemma 1, which can be viewed as a natural extension of Weierstrass theorem. We state it as below.

Lemma 2 *Let* φ *be a nonconstant, bounded, and monotone-increasing continuous function. Let* I_d *denote the d-dimensional unit hypercube* $[0, 1]^d$. *The space of continuous functions on* I_d *is denoted by* $C(I_d)$. *Then, given any function* $f \in C(I_d)$ *and* $\varepsilon > 0$, *there exist an integer n and a set of real constant* α_i, b_i *and vector* $\boldsymbol{a}_i$, *where* $i = 1, 2, \ldots, n$ *such that a function* F *can be defined*

$$F(\boldsymbol{x}) = \sum_{i=1}^{n} \alpha_i \varphi(\boldsymbol{a}_i^T\boldsymbol{x} + b_i) \tag{11}$$

as an approximate realization of the function f; *that is,*

$$|F(\boldsymbol{x}) - f(\boldsymbol{x})| < \varepsilon \tag{12}$$

for all $\boldsymbol{x}$ *that lie in the d-dimensional input space.*

From the Lemma 2, we know that Weierstrass theorem can be naturally extended from *1*-dimensional to *d*-dimensional case. Moreover, the function φ in Lemma 2 can be viewed as a nonlinear input–output mapping. We note that the logistic function (Eq. 2) used as the nonlinearity in a neuronal model for the construction of a MLP is indeed a nonconstant, bounded, and monotone-increasing function, therefore it satisfies the conditions imposed on the function φ in Lemma 2. That is, the universal approximation theorem can be directly applied to MLP.

Another famous network model is the RBFN, which is also capable of forming an arbitrarily close approximation to any continuous functions. This approximation theorem is firstly proposed by Park and Sandberg [9] and stated in Lemma 3 [28].

Lemma 3 *Let* $f : \mathbb{R}^d \to \mathbb{R}$ *be a radial symmetric, integrable, bounded function such that* f *is continuous almost everywhere and* $\int_{\mathbb{R}^d} f(\boldsymbol{x})\mathrm{d}\boldsymbol{x} \neq 0$, *then the family*

$$\sum_{i=1}^{c} \alpha_i g\left(\frac{||\boldsymbol{x} - \boldsymbol{v}_i||_{\mathbb{R}}^d}{2\sigma^2}\right) \tag{13}$$

is dense in $L^p(\mathbb{R}^d)$ *for every* $p \in [1, +\infty)$, *where* $f(\boldsymbol{x}) = g(||\boldsymbol{x}||_{\mathbb{R}})$. Here $L^p(\mathbb{R}^d)$ denotes the usual space of

real-valued maps f defined on $\mathbb{R}^d$ such that f is pth power integrable, and $||.||_{\mathbb{R}^d}$ denotes a distance metric in the space $\mathbb{R}^d$. With the help of Lemma 3, a slight modified version of Lemma 3 addresses that the function f can be locally approximated by an RBFN. We state it in Lemma 4

Lemma 4 *Let $f : \mathbb{R}^d \to \mathbb{R}$ be a integrable bounded function such that f is continuous almost everywhere and $\int_{\mathbb{R}^d} f(\boldsymbol{x})\mathrm{d}\boldsymbol{x} \neq 0$, then the family (Eq. 13) is dense in $L^p_{\mathrm{loc}}(\mathbb{R}^d)$ for every $p \in [1, +\infty)$.*

Here, $L^p_{\mathrm{loc}}(\mathbb{R}^d)$ is a locally-L^pspace and is defined as the set of all measurable functions f such that $f \cdot 1_{[-N,N]^r} \in L^p(\mathbb{R}^d)$ for every $N \in N$. $\mathbf{1}_{[-N,N]^d}$ denotes the characteristic function of a Lebesgue measurable hypertube (subset) $[-N, N]^d$of $\mathbb{R}^d$ and $\mathbb{N}$ denotes the set of natural numbers. Lemma 3 and 4 describe the approximation of an RBFN with respect to the L^p metric or a metric induced from L^p metric. From the proof of Lemmas 3 and 4 [9], we know that there is no requirement of radial symmetry of the function f, that is, those theories are stronger than necessary for RBF networks. Thus, the approximation of RBFNs can be directly induced from the above lemmas.

The foresaid lemmas proof that both MLPs and RBFNs are universal approximators. In the view of approximation, MLPs construct global approximations to nonlinear input–output mapping, while RBFNs construct local approximations to such mapping.

In this section, what we indeed concern is the approximation property of our proposed PGF networks. Some proofs are described as follow.

In the following description, we still use the above notation and definitions. Let $M(I_d)$ denote the finite signed regular Lebesgue measures on I_d. Similar to Ref. [6, 9], we consider only *1*-dimensional output space, and it is trivial to extend *1*-dimensional result to multidimensional output space.

The family of PGF networks consists of the following functions represented by

$$p(\boldsymbol{x}) = \sum_{j=1}^{c} \alpha_j \boldsymbol{\Phi}_j^{\mathrm{PG}}(\boldsymbol{x}) = \sum_{j=1}^{c} \alpha_j \exp\left(-\frac{\left(\boldsymbol{w}_j^T\boldsymbol{x} - \gamma_j\right)^2}{2\sigma_j^2}\right) \tag{14}$$

which are dense in $C(I_d)$ with respect to the supremum norm (L^1 metric), where c is the number of neurons in hidden layer.

For simplicity, let all smoothing factors (widths) in (6) be same, i.e. $\sigma_1 = \cdots \sigma_c = \sigma$ and first review the following definitions.

Definition 2 A function with the form $\phi(\boldsymbol{w}^T\boldsymbol{x} - \gamma)$ is *discriminatory* if for a measure $\mu \in M(I_d)$ satisfies $\int_{I_d} \phi(\boldsymbol{w}^T\boldsymbol{x} - \gamma)\mathrm{d}\mu(\boldsymbol{x}) = 0$ for all $\boldsymbol{w} \in \mathbb{R}^d$ and $\gamma \in \mathbb{R}$ implies that $\mu = 0$.

Obviously, according to the above definition, the function $\exp\left(-\frac{(\boldsymbol{w}^T\boldsymbol{x}-\gamma)^2}{\sigma^2}\right)$ is discriminatory, which is very useful for describing the following approximation of $p(\boldsymbol{x})$.

Theorem 1 *The function $p_\sigma(\boldsymbol{x}) = \sum_{j=1}^{c} \alpha_j \exp\left(-\frac{\left(\boldsymbol{w}_j^T\boldsymbol{x}-\gamma_j\right)^2}{2\sigma^2}\right)$ is dense in $C(\boldsymbol{I}_d)$. That is, for any $f \in C(\boldsymbol{I}_d)$ and $\varepsilon > 0$, there is $|p(\boldsymbol{x}) - f(\boldsymbol{x})| < \varepsilon$, for all $\boldsymbol{x} \in \boldsymbol{I}_d$.*

Proof Let the symbol $\boldsymbol{S}$ be the set of functions of the form $p_\sigma(\boldsymbol{x})$ and $\boldsymbol{S} \subset C(\boldsymbol{I}_n)$. Clearly, $\boldsymbol{S}$ is a linear subspace of $C(\boldsymbol{I}_d)$. What we want to claim is that the closure of $\boldsymbol{S}$, denoted by $\bar{\boldsymbol{S}}$, satisfies $\bar{\boldsymbol{S}} = C(\boldsymbol{I}_d)$.

Assume $\bar{\boldsymbol{S}} \neq C(\boldsymbol{I}_n)$. Then $\bar{\boldsymbol{S}} \subset C(\boldsymbol{I}_n)$ and $\bar{\boldsymbol{S}}$ is still a subspace of $C(\boldsymbol{I}_d)$. By the Hahn–Banach theorem [29] and Lemma 2, there exists a bounded linear function $L(L \neq 0)$ on $C(\boldsymbol{I}_d)$ with the property that $L(\boldsymbol{S}) = L(\bar{\boldsymbol{S}}) = 0$. From Riesz representation theorem [29], we know that L can be described as the form $L(g) = \int_{I_d} g(\boldsymbol{x})\mathrm{d}\mu(\boldsymbol{x})$ for some $\mu \in M(\boldsymbol{I}_d)$ and for all $g \in C(\boldsymbol{I}_d)$. Particularly, since $p_\sigma(\boldsymbol{x}) \in \boldsymbol{S} \subset \bar{\boldsymbol{S}}$ for all $\boldsymbol{w} \in \mathbb{R}^d$ and $\gamma \in \mathbb{R}$, we have that $\int_{I_d} \exp\left(-\frac{(\boldsymbol{w}^T\boldsymbol{x}-\gamma)^2}{2\sigma^2}\right)\mathrm{d}\mu(\boldsymbol{x}) = 0$ for all $\boldsymbol{w}$ and γ. However, as described before, the function $\exp\left(-\frac{(\boldsymbol{w}^T\boldsymbol{x}-\gamma)^2}{2\sigma^2}\right)$ is discriminatory, so it implies that $\mu = 0$, which contradicts the assume $\bar{\boldsymbol{S}} \neq C(\boldsymbol{I}_n)$. Therefore, the subspace must be dense in $C(\boldsymbol{I}_d)$. The proof is complete. □

For $p(\boldsymbol{x})$ defined as (14), the similar approximation can be induced from Theorem 1. We describe this as the following corollary.

Corollary 1 *The function $p(\boldsymbol{x}) = \sum_{j=1}^{c} \alpha_j \exp\left(-\frac{(\boldsymbol{w}_j^T\boldsymbol{x}-\gamma_j)^2}{2\sigma_j^2}\right)$ is dense in $C(\boldsymbol{I}_n)$. For any $f \in C(\boldsymbol{I}_d)$ and $\varepsilon > 0$, there is $|p(\boldsymbol{x}) - f(\boldsymbol{x})| < \varepsilon$, for all $\boldsymbol{x} \in \boldsymbol{I}_d$.*

Some remarks on Theorem 1 and Corollary 1:

Remark 1 The universal approximation can be easily generalized to multiple hidden layers and output nodes PGF networks as described in [6, 9].

Remark 2 For the aforementioned $p_\sigma(\boldsymbol{x})$ and $p(\boldsymbol{x})$, their weighted vectors $\boldsymbol{w}$ has no any constraints except the length.

On the other hand, we can rewrite the (14) as

$$p'(\boldsymbol{x}) = \sum_{j=1}^{c} \alpha_j \exp\left(-\frac{\left\|\boldsymbol{w}_j^T\boldsymbol{x} - \gamma_j\right\|}{2\sigma^2}\right) \quad (15)$$

where $||.||$ denotes a 2-norm distance metric.

Under the constraint $\|\boldsymbol{w}_i\| = 1$, $\left\|\boldsymbol{w}_i^T\boldsymbol{x} - \gamma_i\right\|$ means the distance between the given point $\boldsymbol{x}$ and the ith plane $(\boldsymbol{w}_i, \gamma_i)$. So, the similar local approximation can be directly induced from Lemmas 3 and 4. we state it in the following theorems.

Theorem 2 *Let $f : \mathbb{R}^d \to \mathbb{R}$ be a integrable bounded function such that f is continuous almost everywhere and $\int_{\mathbb{R}^d} f(\boldsymbol{x})\mathrm{d}\boldsymbol{x} \neq 0$, then the family (Eq. 15) is dense in $L^p_{\mathrm{loc}}(\mathbb{R}^d)$ for every $p \in [1, +\infty)$, where $f(\boldsymbol{x}) = g(\|\boldsymbol{x}\|_{\mathbb{R}^d}) = \exp\left(-\frac{\left\|\boldsymbol{w}_i^T\boldsymbol{x} - \gamma_i\right\|}{2\sigma^2}\right)$.*

The following is a direct corollary of Theorem 2.

Corollary 2 *The function $p'(\boldsymbol{x}) = \sum_{j=1}^{c} \alpha_j \exp\left(-\frac{\left\|\boldsymbol{w}_j^T\boldsymbol{x} - \gamma_j\right\|}{2\sigma_j^2}\right)$ is dense in $L^p_{\mathrm{loc}}(\mathbb{R}^d)$ for every $p \in [1, +\infty)$.*

Theorems 1 and 2 describe the approximation of our proposed PGFN with respect to L^1 and L^p metric, respectively. Especially, when $p = 1$ in Theorem 2, the function $f \in L^1$ over a Lebesgue measurable set A can be written as $\int_A f(\boldsymbol{x})\mathrm{d}\boldsymbol{x}$. This is consistent with Theorem 1 when let the set A to be a countable set.

Next we relate our PGFNs to MLPs and RBFNs.

3.4 Relation to other networks

Without loss of generality, the following discussions only consider single-hidden-layer network architecture. In this section, we compare our proposed PGF networks to the two popular networks: MLPs and RBFNs. First of all, let us concern the relation of MLPs and PGFNs.

3.4.1 Relation to MLPs

We discuss their properties mainly including activation function, learning time, and geometrical interpretation. From the viewpoint of constructing activation functions, MLPs usually take the sigmoidal functions as their activation functions (e.g. logistic function, Eq. 2). We rewrite (2) as

$$\Phi_i^S(\boldsymbol{x}) = \frac{1}{1 + \exp(-\boldsymbol{w}_i^T\boldsymbol{x} - \alpha_i)} \quad (16)$$

where $\Phi_i^S(\boldsymbol{x})$ is a output of the ith hidden neuron, $\boldsymbol{w}_i$ means the ith column of linking weight vector between the input and the hidden layer, and α_i is a threshold corresponding to $\boldsymbol{w}_i$.

For the expression $\Phi_i^{\mathrm{PG}}(\boldsymbol{x}) = \exp\left(-\frac{(\boldsymbol{w}_i^T\boldsymbol{x} - \gamma_i)^2}{2\sigma_i^2}\right)$ of the PGF, the parameter pair $\boldsymbol{w}_i$ and γ_i induced from kPC algorithm can also be viewed as the weight and threshold between the input and the hidden layers, respectively. Furthermore, the hidden and output layers of both MLP and PGFN used as a pattern classifier are usually all nonlinear. With the help of definition 2, we know that both activation functions are all discriminatory with the same form $\phi(\boldsymbol{w}^T\boldsymbol{x} - \gamma)$.

From the viewpoint of learning time, however, the PGFNs and MLPs are quite different. Commonly, MLPs adopt BP algorithm to iteratively compute the linking weights between neighboring layers backwards. While for the PGFNs, the weights between the input and the hidden layer are obtained by the kPC algorithm and the weights linking the hidden and output layers can be directly determined by pseudo-inverse method. Therefore, learning time of the PGFNs is far less than that of MLPs.

From the viewpoint of geometrical meaning, to the best of our knowledge, it still unclear how to interpret the geometrical principle of MLP networks. But for the PGFNs, they have clear geometrical interpretation that will be shown in next subsection.

3.4.2 Relation to RBFNs

For PGFNs and RBFNs, although they adopt different activation functions, they have many similar characteristics in both network architecture and weight computation between the hidden and output layers. Detailedly, the activation function of the ith hidden neuron of RBFN is usually presented as $\Phi_i^G(\boldsymbol{x}) = \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{v}_i\|^2}{2\sigma_i^2}\right)$. The value of $\Phi_i^G(\boldsymbol{x})$ corresponds to the output of the ith hidden neuron, $\boldsymbol{v}_i$ is the center position of the function $\Phi_i^G(\boldsymbol{x})$ and σ_i is width of the ith center which affects the generalization capability of that neuron. The c centers, corresponding to c activation functions, usually are computed by FCM algorithm. Figure 2 illustrates a spheral-region (called the receptive field of the Gaussian RBF) determined by the center $\boldsymbol{v}_i$ and width σ_i (radius). The receptive field of RBF is that particular subset of the domain of the input vector $\boldsymbol{x}$ for which $\Phi_i^G(\boldsymbol{x})$ takes sufficiently large values.

While for the PGFNs, they take the PGF as activation function with the expression $\Phi_i^{\mathrm{PG}}(\boldsymbol{x}) = \exp\left(-\frac{\left|\boldsymbol{w}_i^T\boldsymbol{x} - \gamma_i\right|^2}{2\sigma_i^2}\right)$, where σ_i denotes a bandwidth. Particularly, the "centers" of PGF have been changed into central planes, rather than central points of RBF. The tuples $(\boldsymbol{w}_i, \gamma_i)$ are prototypes of a series of plane clusters generating from kPC algorithm. Similarly, from Fig. 6, the band-shaped region can be viewed as the PGF receptive field, where the ith central

plane can be represented as a tuple of $(\boldsymbol{w}_i, \gamma_i)$ and σ_i is a semi-bandwidth.

Summarily, in the aspect of constructing activation function, the PGF is similar to the MLP. But in the aspect of learning method, the PGFN is similar to the RBFN i.e. both are trained by an unsupervised manner. In term of geometrical meaning, both PGFN and RBFN have clear interpretation (Figs. 2, 6). In addition, the PGF forms a semi-closed band-shaped receptive field, which leads PGFN to be capable of showing two-side characteristics: locality and globality. That is, in the direction orthogonal to the central plane $(\boldsymbol{w}_i, \gamma_i)$, the PGF is bounded with the spread σ_i, while in the direction parallel to the $(\boldsymbol{w}_i, \gamma_i)$, it is free. The experimental performances of the classifiers trained by PGFN will be analyzed and discussed in Sect. 4.

4 Experiments

In this section, the performance of the PGFN will be analyzed and compared with the two popular neural networks with one hidden layer: MLPs and RBFNs. All experiments are carried out on three synthetic toy problems and some UCI [30] real-world datasets. In this paper, each toy dataset was equally divided into two parts randomly: training set and test set, while for UCI dataset, tenfold cross-validation was adopted to determine network parameters and test accuracies. To avoid the difference of value, the inputs of all samples are normalized to [0,1]. Computational time was obtained on a machine running Matlab 7.0 on Windows xp with a Pertium IV 2.5 GHz processor and 2G memory.

For classification tasks, many researchers [24, 31–35] concluded that the number of hidden neurons is related to the number of classes, the scale of weights, the number of input samples, the region of the distribution, and so on. However, it is very hard to determine the sample distribution regions, and shapes [24, 31, 32]. Ref. [24] gave us an empirical initial number of hidden nodes as formula $\lfloor 2\log_2(d+c-1) \rfloor$ (where d denotes the dimension of input samples, c denotes the number of classes, and $\lfloor \cdot \rfloor$ denotes a floor operation). However, it is loose in many applications as described in [24]. Teoh et al. [35] said that "for an MLP with one hidden layer, if at least $\frac{\binom{N+d}{d}}{d+2}$ hidden units are used, a smooth activation function can only achieve approximation order $N \in \mathbb{N}$ for all functions $f \in C^N$", where d denotes the dimension of input samples. Moreover, the author also concluded that only one hidden layer is needed for linear or quadratic approximation. That is, a three-layer MLP needs at least $d+1$ hidden neurons to achieve second-order approximation and at least one hidden unit to achieve linear approximation. So, considering the foresaid suggestions, we take two initial hidden nodes in our comparisons [1, 36]. Another difficult problem is how to determine the optimal number of hidden nodes. Huang and Babri [33] told us that the number of inputs n must be an upper bound for any bounded nonlinear activation function. Gao and Ji [24], and Huang and Babri [34] adopted singular value decomposition to estimate an appropriate number of hidden nodes for MLPs. Bartlett [37] suggested that we should pay more attention to network weights rather than network size. Due to comparing the three networks and avoiding too much computation, we take the maximum number of hidden nodes as $\lceil \sqrt{n/2} \rceil$ with a compromise way, where $\lceil \cdot \rceil$ means a ceiling operation. Thus, the number of hidden neurons c varies from 2 to $\lceil \sqrt{n/2} \rceil$ and for each c, we report average percentage of classification accuracies and learning time on 20 independent runs across the tenfolds. In order to clearly report the comparisons, we performed paired t tests comparing MLP to PGFN and RBFN to PGFN. The p-value for each test is the probability of the observed or a greater difference assumption of the null hypothesis that there is no difference between test set correctness distributions. Thus, the smaller the p-value, the less likely that the observed difference resulted from identical test set correctness distributions. A typical threshold for p-value is 0.05. For example, the p-value of the computational time test (Table 3) when comparing PGFN and MLP on the Line3 dataset is 0.000 (<0.05), meaning that PGFN and MLP have different training times on this dataset.

In our study, we use k-means to train RBFNs. Both MLPs and RBFNs were implemented using the Netlab neural network software [38]. The widths of Gaussian RBFs were determined from the k-nearest neighbor according to the following formula:

$$\sigma_j = \left(\frac{1}{k} \sum_{i=1}^{k} \left\| \boldsymbol{x}_i^{(j)} - \boldsymbol{c}_j \right\|^2 \right)^{1/2} \tag{17}$$

Table 3 Average training time (s) of three networks

Datasets	MLP p-value	RBFN p-value	PGFN
Line3	0.0126*	0.0017*	**0.0002**
	0.000	0.012	
Plane3	0.0349*	0.0045*	**0.0023**
	0.002	0.035	
NormalData	0.0542	0.0250	**0.0228**
	0.063	0.038	

Least training time are in bold

* Significant difference from PGFN based on p-value less than 0.05

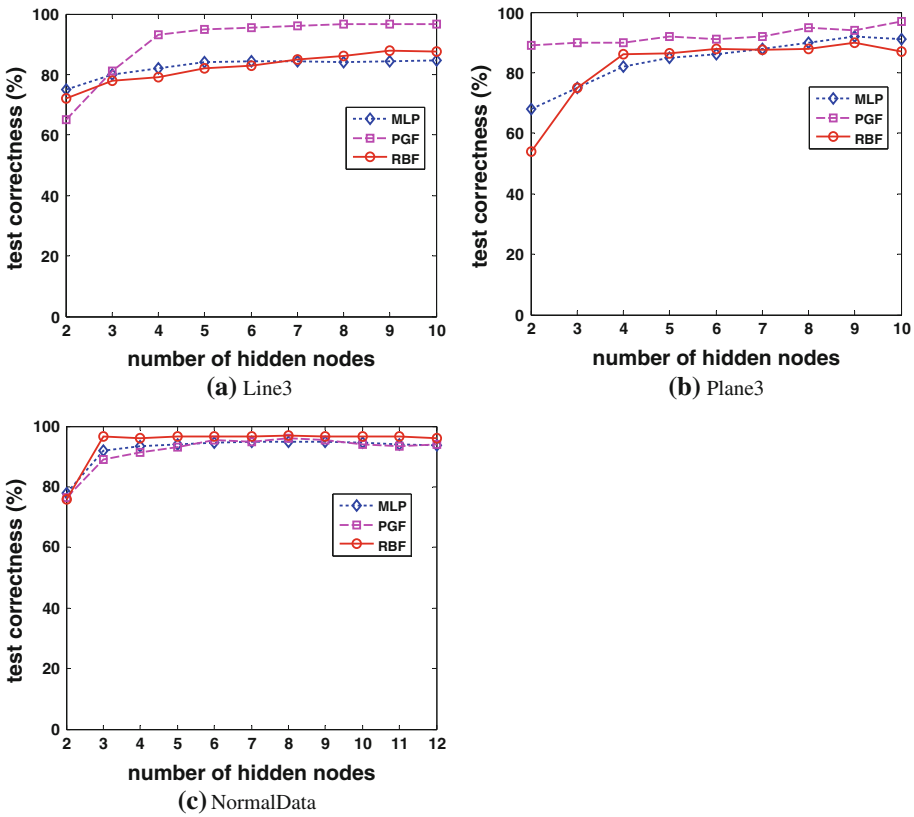

Fig. 8 Tenfolds test accuracies (%) on the three training datasets. **a** Line3, **b** Plane3, **c** NormalData

where $\boldsymbol{c}_j (j = 1, 2, \ldots, c)$ denotes the central vector of the jth cluster, and $\boldsymbol{x}_i^{(j)}$ denotes the ith nearest neighbor of $\boldsymbol{c}_j$.

For the same reason, we define the widths of PGFs as follows under the constraints $\|\boldsymbol{w}_j\| = 1$.

$$\sigma_j = \left(\frac{1}{k_1} \sum_{i=1}^{k_1} \left(\boldsymbol{w}_j^T \boldsymbol{x}_i^{(j)} - \gamma_j \right)^2 \right)^{1/2} \tag{18}$$

where $\boldsymbol{x}_i^{(j)}$ denotes the ith nearest sample to the jth plane $\boldsymbol{w}_j^T \boldsymbol{x} - \gamma_j = 0$. Considering the previous suggestion [39] we set $k_1 = 3$.[1]

Next, we first come to our toy problems.

4.1 Toy problems

Two toy problems were designed on the three-class data (subspace distribution) termed as Line3 (Fig. 4a) and Plane3 (Fig. 4b), respectively. The other one is called NormalData (Fig. 3). The datasets, Line3 (41 samples) and Plane3 (total 300 samples, each class consists of 100 points), were manually generated to simulate three line-shaped and plane-shaped distributions, respectively. The dataset NormalData consists of 2-dimensional samples from three different Gaussian distributions, each of which contains 100 points sampled from its corresponding distribution (see Fig. 3). Their means and covariance are, respectively, represented as $\left(\begin{bmatrix} -0.3 \\ 0 \end{bmatrix}, \begin{bmatrix} 0.16 & \\ & 0.16 \end{bmatrix} \right)$, $\left(\begin{bmatrix} 1 \\ 0.7 \end{bmatrix}, \begin{bmatrix} 0.04 & \\ & 0.09 \end{bmatrix} \right)$ and $\left(\begin{bmatrix} 0 \\ 2 \end{bmatrix}, \begin{bmatrix} 0.36 & \\ & 0.36 \end{bmatrix} \right)$.

Table 3 reports the average training time of three networks on toy problems. Compared to MLP and RBFN, PGFN is significantly outperforming them on datasets Line3 and Plane3. While on NormalData, the training time of three networks is comparable. Figure 8 shows the test accuracies on the training datasets (labeled as "test correctness" in the figures) of the three networks. On the subspace-distributed datasets Line3 and Plane3, the performance of PGFN significantly surpasses MLP and RBFN (Fig. 8a, b). While on the sphere-distributed dataset NormalData, RBFN obtains higher test accuracy than both PGFN and MLP even if in the case of less hidden nodes (Fig. 8c). Incidentally, in order to clearly compare three networks on Line3, we set the upper bound of hidden neurons 10 instead of $\left\lceil \sqrt{41/2} \right\rceil$. On the other hand, Fig. 8 also reports that the test accuracy of PGFN is more stable in tuning the number of the hidden nodes than that of RBFN and MLP. For the reason, it may be that the kPC incorporates the piecewise linear regression into

[1] To distinguish with the foresaid parameter k such as kPC and k-means, here let k_1 denotes the number of nearest neighbors to a given sample.

constructing cluster algorithm under piecewise linear function approximation principle [40]. When the number of the hidden neurons increases, all three networks indicate their favorable performance, which is quite consist with their approximation theories. As described in NFL Theorem, when combining prior knowledge of data distribution, PGF and RBF networks are able to show their superiority to MLP. Figure 4 says that PGFN is more suitable to subspace-distributed datasets Line3 and Plane3, while RBF is suitable to sphere-distributed NormalData.

Next, we validate the performance of three networks on the public UCI datasets.

4.2 UCI datasets

In this section, the performance of three networks were compared in both training time and test accuracies on 13 public UCI datasets. The main information about those datasets is listed in Table 4. Similar to toy problems, we also added p-value (set confidence level 95%) to show statistical difference. In a RBFN, three types of parameters need to be chosen to adapt the network for a given classification task: the central position vectors, the output weights, and the RBF widths. Similarly, PGFNs also need to choose such three types of parameters: central hyperplane, the semi-width, and output weights. As foresaid, the centers of RBFN and PGFN are determined by kPC and k-means algorithms, respectively, here k is a tuning parameter (the number of hidden nodes) varying from 2 to $\left\lceil \sqrt{n/2} \right\rceil$. Then, the widths can be determined by (17) and (18). Since MLPs using a backpropagation are the standard algorithm for any supervised-learning pattern recognition processes and the subject of ongoing research in computational neuroscience; in this section, the learning of MLP is also carried out through BP algorithm.

Table 4 Main information about UCI datasets

Datasets	Number of samples	Dimension	Number of classes
Water	116	38	2 [65, 51]*
Wine	178	13	3 [59, 71, 48]
Bupa	345	6	2 [145, 200]
Sonar	208	60	2 [97, 111]
Iris	150	4	3 [50, 50, 50]
New_thyroid	215	5	3 [150, 35, 30]
Balance_scale	625	4	3 [49, 288, 288]
Ionosphere	351	34	2 [255, 126]
Pima	768	8	2 [500, 268]
Glass	214	9	6 [70,76,17,29,13,9]
Cmc	1473	8	2 [109, 1364]
Waveform	5000	21	3 [1657, 1647, 1696]
WDBC	569	30	2 [212, 357]

* The number in the square brackets is the size of the corresponding class

Table 5 lists our experimental results on average test accuracies (TA) and training time over 20 independent runs. We added a standard deviation (SD) term to TA to measure scatter corresponding test accuracy. Moreover, for each average value of test accuracy and training time, a student's paired t test was applied to examine statistical significance of performance made by PGFN against both MLP and RBFN. Table 5 shows that each dataset contains 4 rows. For example, the first row stands for the maximum test accuracies (%) over tenfold with the appropriate number of hidden nodes, and the second stands for showing their corresponding p-values. When the p-values are less than 0.05, it means that there exists statistical significance between PGFN and the given networks in test accuracies. Similarly, the third and fourth row show average training time and their corresponding p-values, respectively. In 13 UCI datasets, four of them in Table 5 report that PGFNs yield highest test accuracies, and the other five of them, i.e. New_thyroid, Ionosphere, Glass, Cmc, and WDBC, report that there exist statistical insignificance in test accuracies among three networks. The rest datasets report that PGFNs show their insignificances against RBFNs or MLPs. But for training time, the UCI datasets all report that PGFNs are significantly superior to MLPs, and only two of them report that there exist statistical significance between PGFNs and RBFNs.

In general, Tables 3 and 5 say that the training time of PGFNs is significantly less than that of MLPs on both synthetic and UCI benchmark datasets and is comparable to that of RBFNs in the vast majority of datasets. As far as classification performance is concerned, to three types of artificial networks, each of them has its advantages in classification performance. Table 5 also gives such conclusions. Table 3 says that PGFN is more suitable to classify those subspace-distributed datasets, while RBFN is suitable for sphere-distributed datasets.

5 Conclusion

In this paper, we propose a type of novel Plane-Gaussian function networks to enlarge the arsenal of the neural networks. Due to taking the Plane-Gaussian functions as their activation functions, PGFN is capable of approximate any continuous function to arbitrary precision, which is proved in the case of one hidden layer network architecture. Instead of central point in RBF with central plane generated from kPC algorithm, PGF forms a band-shaped receptive field that leads such activation function yield a special geometrical interpretation: locality and globality.

Table 5 Tenfold test accuracies, standard deviation (TA ± SD), average training time (second), and *p*-values corresponding to TA and training time, respectively, over 20 independent runs

Datasets	MLP A ± SD (%) *p*-value of TA Training time (s) *p*-value of time	RBFN TA ± SD (%) *p*-value of TA Training time (s) *p*-value of time	PGFN TA ± SD (%) – Training time (s) –
Water	52.83 ± 1.22*	91.11 ± 4.58	**92.41 ± 3.45**
	0.0000	0.3204	
	0.0498*	**0.0003**	**0.0003**
	0.0000	1.0000	
Wine	**95.01 ± 2.69***	93.41 ± 3.44	91.61 ± 3.64
	0.0001	0.0549	–
	0.0625*	**0.0004**	**0.0004**
	0.0000	1.0000	–
Bupa	70.38 ± 2.77*	73.06 ± 5.33	**75.48 ± 3.55**
	0.0004	0.1356	
	0.0793*	**0.0014***	0.0023
	0.0000	0.0027	
Sonar	**84.72 ± 1.67***	74.83 ± 3.33	74.87 ± 2.59
	0.0012	0.9821	
	0.0635*	**0.0016**	0.0021
	0.0000	0.5681	
Iris	**97.50 ± 1.25***	96.25 ± 1.69*	84.27 ± 12.12
	0.0004	0.0001	
	0.0365*	0.0004	**0.0002**
	0.0000	0.6036	
New thyroid	93.84 ± 4.21	**95.48 ± 3.53**	95.38 ± 2.82
	0.1750	0.5596	
	0.0659*	0.0012	**0.0011**
	0.0000	0.6727	
Balance scale	86.98 ± 3.68*	**90.93 ±** 2.96	90.26 ± 2.98
	0.0022	0.3221	
	0.0795*	**0.0033**	0.0040
	0.0000	0.0634	
Ionosphere	**87.33 ± 1.64**	84.55 ± 1.68	83.24 ± 4.99
	0.2353	0.4644	
	0.0356*	**0.0009***	0.0046
	0.0001	0.0040	
Pima	71.27 ± 1.89*	72.70 ± 3.27*	**76.04 ±** 3.18
	0.0001	0.0367	
	0.0782*	**0.0303**	0.0383
	0.0388	0.2957	
Glass	67.01 ± 9.72	**69.01 ± 4.10**	68.84 ± 4.41
	0.6697	0.8663	
	0.0752*	**0.0004**	0.0007
	0.0000	0.3765	
Cmc	43.49 ± 9.08*	**56.49 ± 3.28**	55.47 ± 3.86
	0.0000	0.1096	
	0.0898*	0.0066*	**0.0023**
	0.0000	0.0475	
Waveform	78.91 ± 2.14*	84.08 ± 8.10	**86.14 ±** 7.58
	0.0025	0.0827	
	1.1360*	0.0538	**0.0501**
	0.0000	0.8146	
WDBC	**97.25 ± 1.15**	95.46 ± 4.22	93.75 ± 6.60
	0.1002	0.1474	
	0.0625*	0.0030	**0.0028**
	0.0000	0.1432	

Highest test accuracies and least training time are in bold

* Significant difference from PGFN based on a *p*-value less than 0.05

Thus, PGFN plays an important role in bridging the gap between the other two types of network: RBFN and MLP.

Experimentally, PGFN runs significantly faster than MLP since PGFN combine the capability of RBF's fast learning property into constructing network. Moreover, with help of *k*-plane clustering algorithm, PGF networks are more suitable for classifying the data with subspace-distributed characteristic.

Here, we should point out that this paper merely concentrates on the description of three basic kinds of networks. How to further optimize the central prototypes and parameters of PGFNs and RBFNs was not concerned in this paper. Intuitively, better classification results can more likely be achieved if some already-improved training algorithms for both RBFNs and MLPs are adopted.

Acknowledgments We thank the anonymous reviewers for their valuable comments and suggestions. We are grateful to the Neural Computing Research Group of Aston university for allowing us to freely use Netlab software. This research was supported by Natural Science Foundation of China (60773061, 60903130), the Jiangsu Science Foundation BK2009393, and Science Foundation of Nanjing Forestry University 163070053 and 163070657.

References

1. Haykin S (1999) Neural networks: a comprehensive foundation, 2nd edn. Prentice Hall, NJ
2. Bishop CM (1995) Neural networks and pattern recognition. Oxford University Press, Oxford
3. Barreto AMS, Barbosa HJC, Ebecken NFF (2006) GOLS-Genetic orthogonal least squares algorithm for training RBF networks. Neurocomputing 69(16–18):2041–2064
4. Sarimveis H, Doganis P, Alexandridis A (2006) A classification technique based on radial basis function neural networks. Adv Eng Softw 37(4):218–221

5. Smyrnakis MG, Evans DJ (2007) Classifying Ischemic events using a Bayesian inference multilayer percetron and input variable evaluation using automatic relevance determination. Comput Cardiol 34:305–308
6. Cybenko G (1989) Approximation by superpositions of a sigmoidal function. Math Control Signals Syst 5(4):303–314
7. Funahashi K (1989) On the approximate realization of continuous mappings by neural networks. Neural Netw 2(3):183–192
8. Hornik K, Stinchcombe M, White H (1990) Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks. Neural Networks 3(5):551–560
9. Park J, Sandberg IW (1991) Universal approximation using radial-basis-function networks. Neural Comput 3(2):246–257
10. Nam MD, Thanh TC (2003) Approximation of function and its derivatives using radial basis function networks. Appl Math Modell 27(3):197–220
11. Lehtokangas M, Saarinen J (1998) Centroid based multilayer perceptron networks. Neural Process Lett 7:101–106
12. Irigoyen E, Pinzolas M (in press) Numerical bounds to assure initial local stability of NARX multilayer perceptrons and radial basis functions. *Neurocomputing*
13. Oliveira ALI, Melo BJM, Meira SRL (2005) Improving constructive training of RBF networks through selective pruning and model selection. Neurocomputing 64:537–541
14. Delogu R, Fanni A, Montisci A (2008) Geometrical synthesis of MLP neural networks. Neurocomputing 71(4–6):919–930
15. De Silva CR, Ranganath S, De Silva LC (2008) Cloud basis function neural network: a modified RBF network architecture for holistic facial expression recognition. Pattern Recogn 41(4): 1241–1253
16. Qu N, Wang L, Zhu M et al (2008) Radial basis function networks combined with genetic algorithm applied to nondestructive determination of compound erythromycin ethylsuccinate powder. Chemom Intell Lab Syst 90(2):145–152
17. Huan HX, Hien DTT, Huynh HT (2007) A novel efficient two-phase algorithm for training interpolation radial basis function networks. Signal Process 87(11):2708–2717
18. Yeung DS, Chan PPK, Ng WWY (2009) Radial basis function network learning using localized generalization error bound. Inf Sci 179:3199–3217
19. Yeung DS, Wang D, Ng WWY, Tsang ECC, Wang X (2007) Structured large margin machines: sensitive to data distributions. Mach Learn 68(2):171–200
20. Duda RO, Hart RE, Stock DG (2001) Pattern classification, 2nd edn. Wiley, New York
21. Bezdek JC (1981) Pattern recognition with fuzzy objective function algorithms. Plenum Press, New York
22. Bradley PS, Mangasarian OL (2000) k-plane clustering. J Global Optim 16(1):23–32
23. Castillo PA, Merelo JJ, Arenas MG, Romero G (2007) Comparing evolutionary hybrid systems for design and optimization of multilayer perceptron structure along training parameters. Inf Sci 177(14):2884–2905
24. Gao D, Ji Y (2005) Classification methodologies of multilayer perceptrons with sigmoid activation functions. Pattern Recognit 38(10):1469–1482
25. Kiernan L, Mason JD, Warwick K (1996) Robust initialization of Gaussian radial basis function networks using partitioned k-means clustering. Electron Lett 32(7): 671–673
26. Bruzzone L, Prieto DF (1999) A technique for the selection of kernel-function parameters in RBF neural networks for classification of remote-sensing images. IEEE Trans Vol Geosci Remote Sensing 37(2):1179–1184
27. Jeffreys H, Jeffreys BS (1988) Methods of mathematical physics, 3rd edn. Cambridge University Press, Cambridge
28. Chen TP, Chen H (1995) Approximation capability to functions of several variables nonlinear functionals and operators by radial basis function neural networks. IEEE Trans Neural Netw 6(4):904–910
29. Rudin W (1987) Real and complex analysis, 3rd edn. McGraw-Hill, Inc., New York
30. Blake C, Keogh E, Merz CJ (1998) UCI repository of machine learning databases [http://www.ics.uci.edu/∼mlearn/MLRepository.html]. Department of Information and Computer Science, University of California, Irvine
31. Draghici S (2002) On the capabilities of neural networks using limited precision weights. Neural Netw 15:395–414
32. Mirchandani G, Cao W (1989) On hidden nodes for neural Nets. IEEE Trans Circuits Syst 36(5):661–664
33. Huang GB, Babri HA (1998) Upper bounds on the number of hidden neurons in feedforward networks with arbitrary bounded nonlinear activation functions. IEEE Trans Neural Netw 9(1): 224–229
34. Teoh EJ, Xiang C, Tan KC (2006) Estimating the number of hidden neurons in a feedforward network using the singular value decomposition. LNCS 3971. Springer, Berlin, pp 858–865
35. Trenn S (2008) Multilayer perceptrons: approximation order and necessary number o hidden units. IEEE Trans Neural Netw 19(5):836–844
36. Mehrabi S, Maghsoudloo M, Arabalibeik H et al (2009) Application of multilayer perceptron and radial basis function neural networks in differentiation between chronic obstructive pulmonary and congestive heart failure diseases. Expert Syst Appl 36:6956–6959
37. Bartlett PL (1998) The sample complexity of pattern classification with neural networks: the size of the weights is more important than the size of the network. IEEE Trans Inf Theory 44(2):525–536
38. Bishop CM, Nabney I (2004) Netlab neural network software. Neural computing research group, Information engineering, Aston University
39. Moody TJ, Darken CJ (1988) Learning with localized receptive fields. In: Hinton G, Sejnowski T, and Touretzsky D (eds) Proceedings of the 1988 connectionist models summer school. Morgan Kaufmann, pp 133–143
40. Bellman RE, Roth RS (1969) Curve fitting by segmented straight lines. Am Stat Assoc J 64:1079–1084