Array 14 (2022) 100161

Contents lists available at ScienceDirect

Array

journal homepage: www.sciencedirect.com/journal/array

Reducing the number of centers in a probabilistic neural network via applying the first neighbor means clustering algorithm

Tetsuya Hoya [*]

Department of Computer Engineering, College of Science and Technology, Nihon University, 7-24-1, Narashino-dai, Funabashi-City, 274-8501, Chiba, Japan

A R T I C L E I N F O

Keywords:
Radial basis function networks
Probabilistic neural network
Center selection
Modified FINCH algorithm
First neighbor means clustering

A B S T R A C T

Probabilistic neural network is a variant of feedforward neural network models and has been successfully applied for various pattern classification purposes. Unlike other feedforward neural network models, probabilistic neural network, a type of radial basis function network models, has essentially only two types of the network-parameters to choose in advance, i.e. the locations of the centers and a single value of radius; a central issue relevant to the application of probabilistic neural network is therefore to determine an appropriate number of the centers accommodated within the network. In the original probabilistic neural network framework, all the training data are allocated to the respective centroid vectors, and thus the network size generally tends to be large, resulting in demanding computational resource. To alleviate this problem, clustering algorithms are commonly employed to shrink the size of the training data. In this work, reduction in the number of centers in a probabilistic neural network is addressed, via the utility of first neighbor means clustering algorithm that is non-iterative and requires only a single algorithmic hyper-parameter; such a choice is desirable in practice. Simulation results using seven publicly available databases for pattern classification tasks show that the first neighbor means clustering algorithm can yield a relatively compact-sized network within short computation time, while exhibiting a reasonably high classification performance, in comparison with communication with local agents, *k*-means, orthogonal least squares, resource allocating network, and resource vector machine algorithms.

1. Introduction

Radial basis function (RBF) networks [1] have been extensively studied in the artificial neural networks community and employed in various scientific fields to date. A typical RBF network has a shallow architecture and is three-layered; units in the first layer equal the respective elements of the tap input vector $\boldsymbol{x}$. Given $\boldsymbol{x}$, each second layer unit yields its node-activation as an RBF

$$h_i(\mathbf{x}) = \exp\left(\frac{-\|\mathbf{x}-\mathbf{t}_i\|_2^2}{\sigma_i^2}\right) \tag{1}$$

where $||\cdot||_2$ denotes $L2$-norm and where $\boldsymbol{t}_i$ and σ_i are called the centroid vector and radius, respectively; (1) implies that the weights between the first- and second-layer units are all unity. The third layer consists of the units, each generating the output as an weighted linear sum of all the M node-activations in the second layer, i.e.

$$o_j = \sum_{i=1}^{M} w_{ij} h_i(\mathbf{x}) \tag{2}$$

Training RBF networks therefore involves, in general, the determination of the respective radii σ_i and weight values between the second- and third-layer units w_{ij}, as well as the assignment of the centroid vectors $\boldsymbol{c}_i$, upon the chosen number of the RBFs, M.

A variant of the RBF networks, probabilistic neural network (PNN) [2] is specifically designed for pattern classification tasks. A PNN has its constraint on the weight values between the second- and third-layer units as binary, i.e., $w_{ij} = 1$, if the i-th second layer unit belongs to Class j, otherwise the value is set to zero. Compared with the ordinary RBF-NNs, this scheme is particularly useful, as no iterative procedure is requied to optimize the weight values, unlike conventional feedforward NN models. Moreover, a unique value for the radii is used in PNN, i.e., in (1), $\sigma_i = \sigma$, the value of which is often chosen heuristically. In the original PNN scheme [2], all the training data are recruited as the respective RBF centers.

[*] .
E-mail address: houya.tetsuya@nihon-u.ac.jp.

https://doi.org/10.1016/j.array.2022.100161
Received 14 September 2021; Received in revised form 15 January 2022; Accepted 1 April 2022
Available online 11 April 2022

1.1. Data clustering algorithms for selecting the centroid vectors

A crucial step for training the RBF-based networks is therefore the choice of the number of the RBFs M and the assignment of the centroid vectors $\boldsymbol{c}_i$. For this aim, data clustering algorithm such as k-means [3] or learning vector quantization (LVQ) [4] is commonly employed. In both the k-means and LVQ, the initial M prototypes are arbitrarily chosen from the training dataset given and then iteratively refined, until a certain condition of convergence is met; such an iterative refinement is carried out in an unsupervised manner in the k-means, whereas the refinement is done in a supervised mode in the LVQ. Hence, the M prototypes obtained upon convergence will be used as the center locations. A recent work [5] proposes a version of the standard k-means algorithm for the improvement over heavily-tailed samples or outliers within the data. Orthogonal least squares (OLS) [6,7] method is another well-known approach used for the determination of the center locations, based upon orthogonal transformation of the training data. In the last two decades, the methods by means of evolutionary computation, such as those based upon particle swarm optimization (PSO) [8] have been proposed for locating the RBF centers [9–11]; the recent work [11] focuses upon the case where the clusters of arbitrary shapes reside in the data structure. The machine learning approach based upon relevance vector machine (RVM) [12] which can be regarded as an enhanced model of support vector machine (SVM) [13], has also been exploited for the purpose of finding the center locations [14]. In Ref. [15], it is also reported that the RVM does not require any additional parameters to tune other than the selection of the kernel function type employed and its associated parameters, while a level of generalization capability comparable to the SVM can be achieved with a fewer number of the kernel functions. In practice, however, the application of the OLS, PSO, or RVM tends to be computationally very expensive, for their relatively high computational complexity, in time and/or memory wise, and/or requirement of non-straightforward tuning of many hyper-parameters. In Ref. [16], another class of clustering algorithm based upon communications with local agents (CLA) is proposed. Like the algorithms based upon density-based spatial clustering of applications with noise (DBSCAN [17,18]), the CLA is a density-based algorithm, motivated by the theory of local gravity in physics. Wang et al. [16] reported that the performance of the CLA algorithm surpassed that of the DBSCAN, while it essentially needs only a single hyper-parameter to be set in advance (i.e., the number of neighbors, whereas the DBSCAN requires two (while the recently proposed Evidential DBSCAN algorithm in Ref. [18] requires three): i.e., distance threshold and the minimum number of points required to form a cluster).

The aim of this work is to determine an appropriate number of RBFs in the second layer of a PNN, while preserving a relatively high classification performance, by applying the first neighbor means (FN-means) clustering algorithm. In the FN-means, different from the clustering algorithms mentioned above, an intuitive approach conforming to original data structure is taken; the first neighbor of each item in the training dataset is considered, and their mutual relations are eventually exploited to determine the RBF center locations. Moreover, unlike the k-means, LVQ, or PSO, the FN-means is non-iterative in nature (i.e., arduous approximation procedure, owing to the repetitive presentation of input data, is not necessary during performing the clustering), while, like the CLA, it requires only a single hyper-parameter to be given in advance, as described next.

1.2. Centroid vectors of RBF networks obtained via data clustering based upon first neighbors

First neighbor[1] [19,20] of an item is defined as the item that yields a minimal distance in between, using a certain distance metric. Then, given a set of N training vectors $\boldsymbol{X} = \{\boldsymbol{x}(1), \boldsymbol{x}(2), \ldots, \boldsymbol{x}(N)\}$, the following relations of the first neighbor indices, i.e. a first neighbor (FN) chain F_c, can be obtained [19]:

$$Fc: i,\ j = FN(i),\ k = FN(j), \ldots, q = FN(p)\ (q \leq N), \tag{3}$$

where the first item in the chain F_c is represented by the i-th vector $\boldsymbol{x}(i)$, which does not appear in any of the previously formed chains (i.e., initially $i = 1$, say). In (3), the relation $j = \mathrm{FN}(i)$ is read as, "the first neighbor of $\boldsymbol{x}(i)$ is $\boldsymbol{x}(j)$," and so forth. The tail of the chain F_c, i.e., the q-th item in (3), is found, if its first neighbor $r = \mathrm{FN}(q)$ appears somewhere in one of the relations in F_c. Note that the chain F_c is not necessarily unicursal, since some of the neighbors in F_c may be identical to each other, i.e.

$$p = FN(m) = FN(n) = \ldots \tag{4}$$

Now, to illustrate how the concept of the first neighbors is exploited to data clustering, let us here examine a crude example: the case where $N = 14$ and a set containing the first-neighbor indices of the respective fourteen vectors S_{FN} is obtained as

$$S_{FN}: \{2, 5, 2, 3, 9, 5, 14, 7, 11, 5, 1, 10, 8, 8\} \tag{5}$$

The set above is interpreted as: the first neighbor of $\boldsymbol{x}(1)$ yields $\boldsymbol{x}(2)$, that of $\boldsymbol{x}(2)$ does $\boldsymbol{x}(5)$, and so forth. The set of the first-neighbor indices given by (5) can then be alternatively represented by the two disjoint directed graphs as depicted in Fig. 1. In the figure, the directed edge $i \rightarrow j$ implies that $j = \mathrm{FN}(i)$ and the relation is nonreversible, i.e., $i \neq \mathrm{FN}(j)$.

Next, extracting only the node numbers (i.e. each representing the index of a training vector) from each of the two graphs depicted in Fig. 1 yields the partition of the original training vector set into the two subsets (or clusters) $\{\boldsymbol{x}(C(1))\}$ and $\{\boldsymbol{x}(C(2))\}$, where the indices of the training vectors $C(1) = \{1,2,3,4,5,6,9,10, 11, 12\}$ and $C(2) = \{7,8,13,14\}$. Therefore, in this example, the means of the vectors in $\{\boldsymbol{x}(C(1))\}$ and $\{\boldsymbol{x}(C(2))\}$ are given as the two representative vectors of the original training vector set, respectively.

This manner of partitioning corresponds to the initial step in order to obtain a first partition (level) of the original dataset within the agglomerative hierarchical clustering operation of the first integer neighbor clustering hierarchy (FINCH) algorithm [20]. In the situations

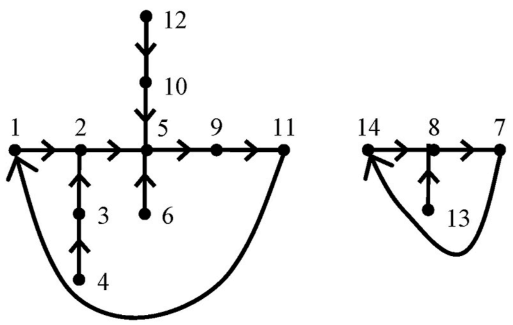

Fig. 1. Two disjoint directed graphs formed by a straight examination of the first neighbors' set given by (5).

[1] The term "first neighbor" is used throughout this work as in [20], instead of "nearest neighbor," in order to emphasize the primacy of the closest one, among the k-nearest neighbors.

where there are more than or equal to two subsets, merging of the subsets (or clusters) will then take place at the subsequent step, and, in due course, another set of clusters, as well as the respective mean vectors, can be obtained at that level of hierarchy. Such an agglomerative clustering step continues, until only a single cluster (i.e., corresponding to the original training dataset, at the top-level of hierarchy) is formed.

In other words, multiple sets of the representative vectors can be eventually obtained at the respective levels of hierarchy. In this work, the representative vectors obtained at a single level of hierarchy are therefore assigned as the respective centroid vectors of an RBF network. Moreover, it is shown that the original FINCH algorithm can be modified to a computationally more efficient form, and hence termed the first neighbor means (FN-means) clustering algorithm, by way of an instantaneous mean computation upon merging of clusters, and applied to determine the RBF center locations, as described next.

2. The first neighbor means clustering algorithm

2.1. Instantaneous mean computation upon merging of clusters into the superset

In an agglomerative step of the original FINCH algorithm [20], except for the first level, the first neighbors are computed using the cluster means $\boldsymbol{m}(i)$ ($i = 1, 2, \ldots, N_{Cl}$; N_{Cl}: num. clusters generated at level l), each obtained by averaging all the $N_{C(i)}$ training vectors in the cluster $C(i)$:

$$\mathbf{m}(i) = \frac{1}{N_{C(i)}} \sum_{j=1}^{N_{C(i)}} \mathbf{x}(C(i,j)). \tag{6}$$

Then, provided that a superset C_s is comprised of the clusters $C(k)$ ($k = 1,2, \ldots, N_{Cs}$; $N_{Cs} \leq N_{Cl}$), the mean of the superset C_s upon a cluster merging $\boldsymbol{m}_{Cs}$ can be computed as

$$\mathbf{m}_{Cs} = \frac{1}{N_{Cs}} \sum_{k=1}^{NCs} \sum_{j=1}^{N_{Cs(k)}} \mathbf{x}(C_s(k,j)) = \frac{1}{N_{Cs}} \sum_{k=1}^{N_{Cs}} N_{Cs(k)} \mathbf{m}(C_s(k)), \qquad N_{Cs} = \sum_{k=1}^{N_{Cs}} N_{Cs(k)}. \tag{7}$$

It is obvious from the relation in the above that the computation of the grand mean $\boldsymbol{m}_{Cs}$ does not need the summing operation for each of the training vectors in the superset, which is redundant, but only that of the cluster means $\boldsymbol{m}(i)$ obtained at the previous level, provided that the number of each cluster $N_{C(i)}$ are preserved.

With this slight modification of using the instantaneous mean computation, the amount of computation in the original FINCH algorithm can be reduced dramatically, as the number of the data vectors becomes large.

2.2. The algorithm

The first neighbor-means (FN-means) clustering algorithm is therefore given as a modified version of the FINCH algorithm, tailored to obtain a set of the representative vectors from the training dataset:

1) (Initialization) Set the maximal level N_L to infinity, if not given explicitly, and the current level of hierarchy $l = 1$.
2) While $l \leq N_L$, repeat the following.

2-1) If $l = 1$, $\boldsymbol{V}$ is set to the N vectors in the original training dataset, i.e., $\boldsymbol{V} = \boldsymbol{X} = \{\boldsymbol{x}(1), \boldsymbol{x}(2), \ldots, \boldsymbol{x}(N)\}$.

- Otherwise, $\boldsymbol{V}$ consists of all the cluster means obtained at the previous level: $\boldsymbol{V} = \boldsymbol{M}(l - 1)$.

2-2) For $\boldsymbol{V}$, obtain the set of the first-neighbor indices S_{FN} as in (5), via constructing the relations in (3). Based upon S_{FN}, form the disjoint graphs as in Fig. 1 (i.e., merging of clusters).

2-3) - If there is formed only a single graph, it reaches a top-level. Terminate.

- Otherwise, each graph corresponds to a single cluster. Obtain each cluster mean $\boldsymbol{m}(i)$ ($i = 1,2, \ldots, N_{Cl}$; N_{Cl}: num. clusters generated at level l) by averaging all the training vectors when $l = 1$ or by applying (7) for $l > 1$. Then, store all the cluster means into the set, i.e., $\mathbf{M}(l) \leftarrow \{\mathbf{m}(i)\}$.

2-4) $l \leftarrow l + 1$.

Given the original training dataset $\boldsymbol{X}$, the FN-means clustering algorithm in the above will generate the cluster means, by default, for all the levels of hierarchy. Therefore, a total of the N_{Cl} cluster means obtained at level l will be assigned as the respective centroid vectors of an RBF network. As the level of hierarchy increases, the number of the cluster means so generated will become smaller, since the algorithm is agglomerative in nature. In other words, it is considered that, with an increasing level of the hierarchy, the data representation will become coarser and thus is considered to yield a poorer performance. Hence, in practice, choosing a level of hierarchy to obtain cluster means, each eventually assigned as the corresponding centroid vector of an RBF $\boldsymbol{t}_i$ in (1), will be dependent upon a trade-off between the performance and the number of the RBFs accommodated within the network.

Nevertheless, it should be emphasized that the FN-means clustering algorithm essentially requires *no* hyper-parameters to be given *a priori*, unlike many clustering algorithms proposed so far, and it automatically determines the number of and in due course generates the representative vectors, each reflecting a certain meaningful aspect of the original data structure. Also, to obtain such representative vectors at a single level of the hierarchy, the FN-means algorithm operates in a non-iterative fashion, unlike many others. Moreover, unlike the k-means, the FN-means is essentially free from the initialization ambiguity; the performance does not vary with the presentation order of the training vectors, since the first neighbor search from any vector in the dataset given will always end up with the same result of the relations as in (3). These properties are quite attractive for the clustering objectives, as the targeted one in this work.

2.3. Class-independent clustering

For the pattern classification tasks to be described next, each of the clustering algorithms is applied separately to each subset of the training dataset for a single class, which is computationally inexpensive, compared to the case where it is applied only once to the entire dataset composed of all the classes. Such a divide-and-conquer clustering strategy as this is also supported by the study [21]; it empirically shows that the network with a lower number of the RBFs found by a class-specific clustering approach can even yield a better classification result, compared to the ordinary case where data clustering is done to the whole training dataset.[2]

3. Simulation study

In the simulation study, five different scenarios of pattern classification tasks were considered, i.e., the pattern classifier based upon 1) the original PNN scheme [2] using a full training dataset, 2) k-nearest neighbors (kNN), 3) resource allocating network (RAN) [22], 4) RVM [12], and 5) PNN with a reduced number of RBFs obtained via the four

[2] Since the LVQ is originally not applicable to the class-specific strategy but whole training data, the clustering scheme is not considered for the simulation study in this work.

clustering algorithms of CLA [16], k-means [3], FN-means, and OLS [6, 7]. The scenarios 1) and 2) were considered as the baseline approaches, while non-PNN but RBF–NN–based approaches of 3) and 4) were also considered for comparison in the simulation study. For 5), the four different clustering algorithms were chosen, since these clustering algorithms are relatively simple to implement with small degrees of freedom for the hyper-parameter selection (i.e., at most two hyper-parameters) and/or operate based upon one-pass presentation of the input data (i.e., non-iterative). Table 1 summarizes the characteristics of these four clustering algorithms in this regard.

3.1. Parameter settings

As summarized in Table 1, both the k-means and OLS require two hyper-parameters; one of the parameters determines the number of the representative (i.e., centroid) vectors for each class in this work, and the other is algorithm-dependent; for the former, the same number of vectors as obtained by the FN-means, given a training dataset, was used. The latter for the k-means is the maximum number of iterations for the convergence (i.e., until there is no change of the centers, the value of which was set to 100 in the simulation study), whereas the width value of each Gaussian function (i.e., RBF) for the OLS, which was set to the same value as the unique radii σ for a PNN in the simulation study. For the CLA, it was observed that the algorithm did not yield no reduction of the training data at all, when the setting of the single parameter, i.e., 1NN for the density estimation, was used; the k (>1) nearest neighbors for the CLA was considered for the simulation study.

In contrast to the four clustering algorithms applied to a PNN, both the RAN and RVM are based upon an ordinary radial basis function network scheme; the weights between the hidden and output layers are not binary but optimized. For both the RAN and RVM, the number of the representative vectors is automatically determined, like the FN-means, once a training dataset is given. For the RVM, three hyper-parameters, i.e., the radius (which was set equal to the unique radii of the PNN σ, as for the OLS) and the maximum number of iterations, as well as that for posterior mode finder (the value of which were set to 1000 and 2, respectively), needs to be set in advance. In addition to the parameter setting, the commonly employed 'one-versus-the-rest' strategy [23] was taken for constructing the multi-class pattern classifiers based upon the RVM. On the other hand, the RAN has six parameters to be given *a priori*, though only one-pass presentation of the dataset is required during the training; for the four out of the six, the same parameters as in Ref. [22] were used for the RAN in the simulation study, i.e., $\alpha = 0.05$, $\delta_{\max} = 0.7$, $\delta_{\min} = 0.07$, and $\kappa = 0.87$. For both the decay constant and the error threshold of the RAN, the settings $\tau = 40$ and $\varepsilon = 0.2$ were respectively found to yield a reasonable classification performance, as well as relatively compact networks, and thus were used for the simulation study.

For the four scenarios 1), 2), 4), and 5) in the above, all the simulations were performed using the unique radii setting:

$$\sigma = \beta d_{\max}/N_c, \tag{8}$$

where $d_{\max}$ is the maximum Euclidean distance calculated over all the pairs of the vectors in the training dataset and N_c the number of classes for a dataset. For the multiplying factor β, the setting $\beta = 1$, was always used for the scenarios 1), 2), and 5), but was varied within the range [2, 8] for 4) (i.e., the RVM scenario) only, so as to avoid the numerical instability (i.e., ill-conditioned Hessian) occurred during training the RVM.

Table 1
Summary of the four clustering algorithms used.

	Num. hyper-parameters	Non-iterative algorithm	Affected by order of presentation
CLA	1	N	Y
k-means	2	N	Y
FN-means	1	Y	N
OLS	2	Y	N

3.2. Datasets used

The seven datasets for image/speech recognition were used for the simulation study, i.e. the datasets using 1) ISOLET, 2) Optical Recognition of Handwritten Digits (OptDigits), 3) Pen-Based Recognition of Handwritten Digits (PenDigits), 4) Letter Recognition (LR), 5) MNIST-10 K (a smaller version of the original MNIST database [24]), 6) Speech Filing System (SFS), and 7) STL-10 database; the four databases 1) - 4) were publicly available from the UCI machine repository [25], whereas 6) and 7) were obtained from Refs. [26,27], respectively; the databases ISOLET and LR contain the pattern vectors of spoken and character image feature of English alphabets, respectively, whereas OptDigits, PenDigits, and MNIST-10 K are the databases of handwritten digits. The SFS is comprised by the feature data of spoken digits in English (i.e., the original SFS database was only available in raw speech format). The STL-10 is a ten-class database consisting of the color images of airplane, bird, car, cat, deer, dog, horse, monkey, ship, and truck. Table 2 summarizes the properties of the seven datasets used in the simulation study. As shown in Table 2, the number, as well as the vector length, of the patterns varied from small to medium large for the datasets.

For all the seven datasets, no other feature extraction but normalization within the range [−1.0, 1.0] was performed for each pattern vector in the datasets, prior to performing the pattern classification tasks.

During the simulation, the divide-and-conquer strategy described in Section 3.3 was introduced to each of the five scenarios; each method was applied independently to the subset of each class for training and thereby was eventually constructed a PNN/RBF network for performing the pattern classification tasks.

3.3. Simulation results

Table 3 summarizes the simulation results obtained for each method using all the seven datasets of ISOLET, LR, OptDigits, PenDigits, MNIST-10 K, SFS, and STL-10, in terms of classification rate (i.e., denoted "C. Rate (%)" in the table), number of RBFs, and the execution time for clustering the training data/constructing a PNN/RBF network, based all upon using Octave scripts ("Time (sec)"; the scripts for each of the methods were coded/arranged to yield a similar level of optimization, as much as possible). For the kNN, the performance with the number of the neighbors k (i.e., varied within the range [1,10]) that yielded the best classification rate is shown in the table.

Table 4 shows the results of the paired t-test (two-tailed), obtained via performing 10-fold cross-validation (CV), in order to verify statistically the classification performance obtained in the simulation study as shown in Table 3; each of the p- and t-value pair was computed for comparing the performance between the FN-means and other methods (except for both the baselines of original PNN and kNN). In the test, if p was less than 0.05 (i.e., the confidence of 95%) and if the modulus of t

Table 2
Summary of the seven datasets used for the simulation study.

	Num. Training Vectors	Num. Testing Vectors	Length of Each Vector	Num. Classes
ISOLET	6238	1559	617	26
LR	16,000	4000	16	26
OptDigits	3823	1797	64	10
PenDigits	7494	3498	16	10
MNIST-10 K	10,000	2000	784	10
SFS	540	360	256	10
STL-10	10,000	3000	2048	10

Table 3
Summary of the simulation results obtained using the seven datasets.

Dataset	Method	C. Rate (%)	Num. RBFs	Exec. Time (sec)	Sig. Diff.	Dataset	Method	C. Rate (%)	Num. RBFs	Exec. Time (sec)	Sig. Diff.
ISOLET	FN-means ($N_L = 1$)	90.89	1924	**76.4**	–	MNIST-10 K	FN-means ($N_L = 1$)	92.95	1951	414.8	–
	CLA (2NN)	91.4	2411	230.3	No		CLA (2NN)	92.75	2957	**389.7**	No
	k-means	91.53	1924	279.3	No		*k*-means	93.1	1951	2582.3	No
	OLS	85.12	1924	254.9	Yes		OLS	87.85	1951	3777.9	Yes
	RAN	**96.02**	5229	799.6	Yes		RAN	**95.85**	7769	1969	Yes
	RVM ($\beta = 10$)	95.9	**602**	2417.6	Yes		RVM ($\beta = 3$)	93.45	**743**	3029.1	Yes
	Orig. PNN	88.84	6238	–	–		Orig. PNN	92.2	10,000	–	–
	9NN	92.37	6238	–	–		3NN	92.1	10,000	–	–
LR	FN-means ($N_L = 1$)	94.83	4180	**456.5**	–	SFS	FN-means ($N_L = 1$)	98.33	107	1.3	–
	CLA (2NN)	95.38	6285	473.7	Yes		CLA (2NN)	98.33	144	2.4	No
	k-means	95.08	4180	2765.7	No		*k*-means	96.11	107	3.4	No
	OLS	88.43	4180	2965.4	Yes		OLS	91.39	107	**0.5**	Yes
	RAN	**96.83**	9986	4224.6	Yes		RAN	**98.06**	516	5.4	Yes
	RVM ($\beta = 3$)	95.55	**1746**	16757.3	Yes		RVM ($\beta = 10$)	95.6	**98**	14.4	No
	Orig. PNN	96.2	16,000	–	–		Orig. PNN	96.94	540	–	–
	1NN	95.58	16,000	–	–		5NN	97.5	540	–	–
OptDigits	FN-means ($N_L = 1$)	97.83	766	**58.9**	–	STL-10	FN-means ($N_L = 1$)	95.97	1151	**357.8**	–
	CLA (2NN)	98.05	1105	101.2	No		CLA (2NN)	96.07	2533	470.5	Yes
	k-means	97.55	766	287.7	No		*k*-means	96.17	1151	2149.4	No
	OLS	96.16	766	170.1	Yes		OLS	94.57	1151	1507.2	Yes
	RAN	98.22	2056	231.5	No		RAN	**97.07**	8350	2285.2	Yes
	RVM ($\beta = 2$)	96.27	**215**	260.7	No		RVM ($\beta = 3$)	96.73	**493**	2935.6	Yes
	Orig. PNN	**98.33**	3823	–	–		Orig. PNN	95.8	10,000	–	–
	1NN	98	3823	–	–		4NN	96.6	10,000	–	–
PenDigits	FN-means ($N_L = 1$)	94.57	1748	251.8	–	PenDigits	RAN	97.34	2251	597.6	Yes
	CLA (2NN)	94.63	2321	**229.6**	No		RVM ($\beta = 2$)	97.34	**185**	1017.8	Yes
	k-means	93.85	1748	1642.5	No		Orig. PNN	94.25	7494	–	–
	OLS	94.74	1748	1872.5	No		3NN	**97.8**	7494	–	–

Table 4
Results of the paired *t*-test compared with FN-means ($N_L = 1$) against other clustering algorithms for the seven datasets.

Dataset	Method to Compare	*t*-value	*p*-value	Sig. Diff.	Dataset	Method to Compare	*t*-value	*p*-value	Sig. Diff.
ISOLET	CLA (2NN)	−1.0464	3.23×10^{-1}	No	MNIST-10 K	CLA (2NN)	−0.152106	8.82×10^{-1}	No
	k-means	−2.1657	5.85×10^{-2}	No		*k*-means	−0.108183	9.16×10^{-1}	No
	OLS	5.7006	2.94×10^{-4}	Yes		OLS	15.642013	7.84×10^{-8}	Yes
	RAN	−9.2671	6.72×10^{-6}	Yes		RAN	−11.578954	1.04×10^{-6}	Yes
	RVM ($\beta = 10$)	−5.6261	3.23×10^{-4}	Yes		RVM ($\beta = 3$)	−3.174055	1.129213e-02	Yes
LR	CLA (2NN)	−3.6766	5.10×10^{-3}	Yes	SFS	CLA (2NN)	−2.1429	6.07×10^{-2}	No
	k-means	−0.6150	5.54×10^{-1}	No		*k*-means	−0.2175	8.33×10^{-1}	No
	OLS	19.9610	9.24×10^{-9}	Yes		OLS	3.5938	5.80×10^{-3}	Yes
	RAN	−17.4668	2.99×10^{-8}	Yes		RAN	−2.4476	3.69×10^{-2}	Yes
	RVM ($\beta = 3$)	−3.9354	3.43×10^{-3}	Yes		RVM ($\beta = 10$)	0.4598	6.57×10^{-1}	No
OptDigits	CLA (2NN)	−0.7943	4.47×10^{-1}	No	STL	CLA (2NN)	−2.4081	3.94×10^{-2}	Yes
	k-means	1.0578	3.18×10^{-1}	No		*k*-means	−0.7245	4.87×10^{-1}	No
	OLS	4.7150	1.10×10^{-3}	Yes		OLS	11.6173	1.01×10^{-6}	Yes
	RAN	−1.3591	2.07×10^{-1}	No		RAN	−5.9772	2.08×10^{-4}	Yes
	RVM ($\beta = 2$)	5.1275	6.22×10^{-4}	No		RVM ($\beta = 3$)	−4.8824	8.68×10^{-4}	Yes
PenDigits	CLA (2NN)	−0.3645	7.24×10^{-1}	No	PenDigits	RAN	−9.7246	4.51×10^{-6}	Yes
	k-means	−0.9756	3.55×10^{-1}	No		RVM ($\beta = 2$)	−8.0306	2.15×10^{-5}	Yes
	OLS	−1.1310	2.87×10^{-1}	No					

was greater than the critical value of 2.262, a statistically significant difference was observed between the two methods, otherwise there was no significant difference between them; the negative/positive value of t indicates that the performance obtained via the FN-means was inferior/superior to the other method in comparison. For convenience, the evaluation of the paired *t*-test also appears in Table 3 (i.e., in the sixth row for each dataset; "Sig. Diff." shows "Yes", when there was a significant difference in between).

3.4. *Discussion of the simulation results in view of the computational complexity of the methods*

As shown in Table 3, it was observed that the numbers of the centroid vectors obtained using the RVM were always smaller than those using the other approaches in the simulation study. However, as indicated by the execution time in Table 3, this is considered to be the consequence of a thorough optimization which leads to a rather expensive computation; the RVM is an iterative algorithm, and each iteration involves the inversion of Hessian matrix which requires $O(N^3)$ computation. Moreover, as described earlier, rather a careful selection of the radius value was needed for avoiding the numerical instability relevant to the inversion of the matrix, during the simulation. In contrast, the FN-means is a non-iterative algorithm, and the computationally heaviest part in the FN-means is the first neighbors search at $l = 1$; the search requires $O(N^2/2)$ computation for all the pairs of the training vectors, which is however much lower than that of the RVM. Although RAN is a one-pass algorithm and the classification rates obtained by the RAN were the highest for all but the PenDigits cases, RAN can be also regarded as a computationally demanding algorithm; at the presentation of each training pattern, the search for the nearest centroid vector, as well as the updating of all the centroid vectors accommodated so far, is performed. Moreover, to seek an optimal combination of the six parameters, as described earlier, is a non-straightforward task, whereas the FN-means in essence does not require any parameter tuning beforehand at all. As shown in Table 3, it was also observed that the size of the RBF network so constructed was almost always the largest among all the methods except those of the baselines i.e., the original PNN and kNN.

The OLS involves the Gram-Schmidt orthogonalization operation, and so it requires $O(N^2M)$ (M: the number of the best regressors, i.e., set equal to the centroid vectors of a PNN in this work) computation [28], as well as $O(N^2)$ storage in order for the full-sized distance matrix to be kept during the computation, whereas the memory space required for the FN-means is at most $O(\log(N)) \sim O(N)$, for storing the cluster mean vectors at each level of hierarchy. In this regard, it is considered that the computation of the OLS yields generally higher than that of the FN-means. Moreover, as in Table 3, it was observed that for most of the cases the classification rates obtained by the FN-means were consistently higher than those obtained using the OLS during the simulation study (as statistically confirmed by the results of the paired t-test).

For the k-means, the computational complexity is said to be $O(NMK)$, where M and K are the number of the prototypes (i.e. the centroid vectors) and that of the iterations till convergence, respectively. It is then said that the amount of computation depends upon the situation; for each of the cases in Table 3, it is considered that $N < 2$ MK, as the execution time was longer than that of the FN-means. Nevertheless, it can be said that the complexity of the k-means was higher than that of the FN-means for these cases. Moreover, it should also be notable that, while the FN-means does not essentially depend upon the presentation order of input data, the classification performance varies to a certain degree for both the k-means and CLA; in the conventional k-means approach, the M prototypes are initially chosen randomly from the original training dataset.

3.5. Comparison with the CLA

In addition to the ambiguity of the classification performance, in the preliminary simulation study, it was observed that the number of the centroid vectors obtained via the CLA algorithm was also varied by the input presentation order; it is considered that this was also due to the inherent nature of the data-driven density estimation. For the complexity, since the CLA exploits the kNN within the algorithm, the computational cost of $O(N^2/2)$ will be at least required, like the FN-means, in addition to the density estimation in the main part of the algorithm, the estimation of which is not as simple as the FN-means. This indicates that the total amount of the computational cost will be generally higher than that of the FN-means. In turn, as shown in.

Table 3, the number of the centroid vectors obtained via the CLA was always greater than that obtained using the FN-means for all the cases, while the execution time by the CLA was sometimes comparable to/shorter but much longer than the FN-means for most of the cases. Moreover, no noticeable discrepancy between the CLA and FN-means was observed in terms of the classification rates, the latter of which can be also confirmed by the results of the p-test as in Table 4.

Nevertheless, it should be highlighted that, like the FN-means, the degree of freedom of the CLA is smaller than the other methods used in the simulation study, i.e., only one; the CLA has a single parameter of k (i.e., for the kNN search) to be determined *a priori*, and, since the number of the centroid vectors obtained was greatly dependent upon this parameter setting during the simulation, another comparison is made; Tables 5–7 compare the numbers of the centroid vectors (i.e., RBFs) for PNN obtained via the CLA and FN-means and the classification rates, with varying k for the CLA whereas the maximum hierarchical level N_L for the FN-means, respectively. In these tables, the simulation results obtained using the CLA shown are those that yielded the numbers of the centroid vectors similar to the FN-means. It is observed that the classification rates obtained using the FN-means are relatively kept higher for the increasing levels of N_L (as confirmed by the paried t-test, as shown in the tables), with, in counterwise, the smaller numbers of the centroid vectors than those of the CLA, for most of the cases. This indicates that the original data structure can be better preserved by the FN-means, in comparison with the CLA.

Table 5
Comparison between the CLA and FN-means for the ISOLET.

ISOLET	Number of RBFs (for FN-means $N_L = 1,2,3$)			
	FN-means	1924	199	55
	CLA	2411 (2NN)	239 (4NN)	58 (9NN)
	Classification Rate (%)			
	FN-means	90.89	**90.31**	**88.97**
	CLA	91.4	89.1	86.02
	Sig. Diff.	No	Yes	Yes

Table 6
Comparison between the CLA and FN-means for the LR, OptDigits, and PenDigits.

LR	Number of RBFs (for FN-means $N_L = 1,2,3$)			
	FN-means	4180	1104	329
	CLA	6285 (2NN)	970 (4NN)	391 (9NN)
	Classification Rate (%)			
	FN-means	94.83	**91.05**	**81.6**
	CLA	**95.38**	81.67	74.85
	Sig. Diff.	Yes	Yes	Yes
OptDigits	Number of RBFs (for FN-means $N_L = 1,2,3$)			
	FN-means	766	161	35
	CLA	1105 (2NN)	104 (4NN)	35 (6NN)
	Classification Rate (%)			
	FN-means	97.83	**96.49**	**92.27**
	CLA	98.05	94.1	90.98
	Sig. Diff.	No	Yes	Yes
PenDigits	Number of RBFs (for FN-means $N_L = 1,2,3$)			
	FN-means	1748	417	100
	CLA	2321 (2NN)	568 (3NN)	116 (5NN)
	Classification Rate (%)			
	FN-means	94.57	94.48	91.48
	CLA	94.63	94.08	**92.25**
	Sig. Diff.	No	No	Yes

Table 7
Comparison between the CLA and FN-means for the MNIST-10 K, SFS, and STL-10.

MNIST-10 K	Number of RBFs (for FN-means $N_L = 1,2,3$)			
	FN-means	1951	338	74
	CLA	2962 (2NN)	369 (4NN)	104 (8NN)
Classification Rate (%)				
	FN-means	92.94	**90.3**	**85.9**
	CLA	92.75	88.2	82.1
	Sig. Diff.	No	Yes	Yes
SFS	Number of RBFs (for FN-means $N_L = 1,2$)			
	FN-means	104	25	
	CLA	145 (2NN)	32 (4NN)	
	Classification Rate (%)			
	FN-means	98.33	95	
	CLA	98.33	94.44	
	Sig. Diff.	No	No	
STL-10	Number of RBFs (for FN-means $N_L = 1,2$)			
	FN-means	1151	79	
	CLA	756 (3NN)	88 (7NN)	
Classification Rate (%)				
	FN-means	95.97	**95**	
	CLA	96	94.77	
	Sig. Diff.	No	Yes	

4. Conclusion

In this work, a novel approach for determining the centers of RBF networks has been proposed, based upon the first neighbor means clustering algorithm, which can be regarded as a modified version of FINCH algorithm [20]. The FN-means is a computationally inexpensive and one-pass algorithm and essentially requires no parameter tuning, unlike many conventional clustering algorithms. As shown in the work, given the training dataset, a compact-sized PNN can be constructed straightforwardly, while automatically assigning the location of the centers by applying the FN-means algorithm. Thereby, a performance comparable to the approach such as CLA, *k*-means, OLS, RAN, or RVM can be obtained, as seen in the simulation study in this work. In the case of the PNN, a unique value of the radius is therefore the only parameter to be prespecified for the case where the FN-means is applied, which is quite handy to implement and hence considered to be attractive. By virtue of the simplicity as well as transparency in structure wise of the PNN, the introduction of the FN-means clustering algorithm to the center selection has a potential to enhance the versatility of RBF type networks in various application areas of interest.

Declaration of interests

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests:

References

[1] Broomhead DS, Lowe D. Multivariate functional interpolation and adaptive networks. Complex Syst 1988;2:321–55.
[2] Specht DF. Probabilistic neural networks. Neural Network 1990;3(1):109–18.
[3] MacQueen JB. Some methods for classification and analysis of multivariate observations. In: Proc. Symp. Matho. Stat. Prob. fifth first ed. Berkeley: Univ. Calif. Press; 1967. p. 281–97.
[4] Kohonen T. Learning vector quantization. In: Kohonen T, editor. Self-Organizing Maps. Springer; 1995. p. 175–89.
[5] Li Y, Zhang Y, Tang Q, Huang W, Jiang Y, Xia S-T. t-k-means: a robust and stable k-means variant. In: ICASSP-2021 - Int. Conf. Acoustic Speech Signal Processing; 2021. p. 3120–4.
[6] Chen S, Cowan CFN, Grant PM. Orthogonal least squares learning algorithm for radial basis function networks. IEEE Trans Neural Network 1991;2(2):302–9.
[7] Chen S, Grant PM, Cowan CFN. Orthogonal least-squares algorithm for training multioutput radial basis function networks. Proc. Inst. Elect. Eng., pt. F 1992;139 (6):378–84.
[8] Kennedy J, Eberhart R. Particle swarm optimization. IEEE Int. Conf. Neural Networks 1995;4:6390–4.
[9] Chen S, Hong X, Harris CJ. Particle swarm optimization aided orthogonal forward regression for united data modeling. IEEE Trans Evol Comput 2010;14(4):477–99.
[10] Deng J, Li K-, Irwin GW, Fei M. Two-stage RBF network based on particle swarm optimization. Trans Inst Meas Control 2013;35(1):25–33.
[11] Luo W, Zhu W, Ni L, Qiao Y, Yuan Y. SCA2: novel efficient swarm clustering algorithm. IEEE Trans. Emerging Topics in Computational Intelligence 2021;5(3): 442–56.
[12] Tipping ME. Sparse Bayesian learning and the relevance vector machine. J Mach Learn Res 2001;1:211–44.
[13] Cortes C, Vapnik V. Support-vector network. Mach Learn 1995;20:273–97.
[14] Liu X, Li R-, Cheng D, Cheng K. Investigation on the construction of the relevance vector machine based on cross entropy minimization. In: ICAC-2016 - 22nd Int. Conf. Automation and Computing; 2016. p. 233–7.
[15] Tipping ME. The relevance machine. In: NIPS-99 - 12th Int. Conf. Neural Information Processing Systems; 1999. p. 652–8.
[16] Wang Z, Yu Z, Chen CLP, You J, Gu T, San Wong H, Zhang J. Clustering by local gravitation. IEEE Trans Cybern 2018;48(5):1383–96.
[17] Ester M, Kriegel H-P, Sander J, Xu X. A density-based algorithm for discovering clusters in large spatial databases with noise. In: 2nd Int. Conf. Knowledge Discovery and Data Mining; 1996. p. 226–31.
[18] Bessrour M, Elouedi Z, Lefevre E, Dbscan E-. An evidential version of the DBSCAN method. In: SSCI-2020 - 2020 IEEE Symp. Series on Computational Intelligence; 2020. p. 3073–80.
[19] Murtagh F. A survey of recent advances in hierarchical clustering algorithms. Comput J 1983;26(4):354–9.
[20] Sarfraz M, Sharma V, Stiefelhagen R. Efficient parameter-free clustering using first neighbor relations. In: IEEE/CVF CVPR-2019 - 2019 IEEE/CVF Conf. Computer Vision and Pattern Recognition; 2019. p. 8926–35.
[21] Raitoharju J, Kiranyaz S, Gabbouj M. Training radial basis function neural networks for classification via class-specific clustering. IEEE Transact Neural Networks Learn Syst 2016;27(12):458–2471.
[22] Platt JC. A resource-allocating network for function interpolation. NIPS-1991 - Advances in Neural Information Processing Systems 1991;9:765–71.
[23] Bishop CM. Pattern recognition and Machine Learning. first ed. Springer; 2006.
[24] LeCun Y, Cortes C, Burges CJC. The MNIST database. http://yann.lecun.com/exdb/mnist/.
[25] Asuncion A, Newman D. UCI machine learning. Irvine, Irvine, CA: Univ. California; 2007. https://archive.ics.uci.edu/ml/datasets.php.
[[26] Huckvale M. Speech filing system vs3.0 – computer tools for speech research. London, U.K.: Univ. College; 1996.
[27] Coates A. https://cs.stanford.edu/~acoates/stl10/.
[28] Golub GH, van Loan CF. Matrix Computations. third ed. John Hopkins; 1996.