FINDING INITIAL PARAMETERS OF NEURAL NETWORK FOR DATA CLUSTERING

Suneetha Chittineni[1] and Raveendra Babu Bhogapathi[2]

[1]R.V.R. & J.C. College of Engineering, Guntur, India.
Chittineni_es@yahoo.com
[2]VNR Vignana Jyothi Institute of Engineering & Technology, Hyderabad, India.
rbhogapathi@yahoo.com

ABSTRACT

K-means fast learning artificial neural network (K-FLANN) algorithm begins with the initialization of two parameters vigilance and tolerance which are the key to get optimal clustering outcome. The optimization task is to change these parameters so a desired mapping between inputs and outputs (clusters) of the K-FLANN is achieved. This study presents finding the behavioral parameters of K-FLANN that yield good clustering performance using an optimization method known as Differential Evolution. DE algorithm is a simple efficient meta-heuristic for global optimization over continuous spaces. The K-FLANN algorithm is modified to select winning neuron (centroid) for a data member in order to improve the matching rate from input to output. The experiments were performed to evaluate the proposed work using machine learning artificial data sets for classification problems and synthetic data sets. The simulation results have revealed that optimization of K-FLANN has given quite promising results in terms of convergence rate and accuracy when compared with other algorithms. Also the comparisons are made between K-FLANN and modified K-FLANN.

KEYWORDS

Optimization, Clustering, Differential Evolution, Artificial neural network

1. INTRODUCTION

Optimization is a procedure used to make a clustering mechanism as effective as possible using mathematical techniques. It is nothing but finding an alternative with the highest achievable performance under the given constraints.

Clustering is the process of organizing objects in to groups, whose members are similar in some way. A cluster is therefore a collection of objects which are similar to each other and dissimilar to the objects belonging to other clusters. In the context of machine learning, clustering is an unsupervised task. The input for clustering problem is the set of objects, where each object is described by its relevant attributes. The desired output is the partitioning set of objects into disjoint groups by minimizing the objective function. Clustering problems are of two types. In hard clustering, an object is assigned to only one cluster. In fuzzy clustering, an object is assigned to more than one cluster with some membership degree.

Artificial neural network composed of inter-connected artificial neurons that mimic the properties of biological neural networks. The major applications of artificial neural networks applied to clustering in different domains are image segmentation for grouping similar images, in data mining dividing data set into different groups, in bio-informatics grouping genes which are having similar characteristics. In this paper, clustering is carried out using Modified k-means fast learning artificial neural network.

Evolution of fast learning artificial neural networks begins with the two basic models proposed by Tay and Evans: FLANN I and FLANN II. FLANN I model is the modification of Adaptive Resonance Theory (ART 1) developed by Carpenter and Grossberg [6]. The design of FLANN II was based on Kohonen LVQ network with Euclidian distance as similarity metric [7].The later improvement in the FLANN II is to take continuous value in input [8]. Later K-means calculations were included in the FLANN algorithm. This leads to the development of K-FLANN [9].Later KFLANN was modified to include data point reshuffling which resolves the data sequence sensitivity that creates stable clusters [10].

The most popular and well known algorithm known as Genetic algorithms, developed by John Holland in 1975, are a model or abstraction of biological evolution based on Charles Darwin's theory of natural selection [2].Later the biological crossover and mutation of chromosomes are simulated in the algorithm by Holland and his collaborators to improve the quality of solutions over successive iterations [3][24].

Differential evolution was first popularized by R. Storn and K. Price in 1997[1]. It is vector based evolutionary algorithm that is highly efficient for global optimization problems [12-15]. It can be considered as further development of genetic algorithm. Storn and Price in 1997 stated that DE was more efficient than simulated annealing and genetic algorithms [1][18][19]. When compared to GA, DE is very attractive due to its simple design and good performance. The implementation of DE is easy because simple mathematical operators can be applied to evolve the population with few control parameters [1]. The most important features of DE noticed by several researchers are [11] fast convergence, speed, stronger stability, easy to realize and easily adaptable for integer and discrete optimization. The same work is carried with GA also. When GA is combined with these algorithms, it was noticed that GA is efficient for data sets of small size .But when size of the data set is large; GA requires more execution time than DE to perform data clustering [24].

In DE, all solutions have the same chance of being selected as parents without dependence on their fitness value. The better one of new solution and its parent wins the competition providing significant advantage of converging performance over genetic algorithms (GA) [11].

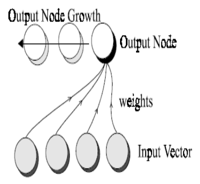
2. K-MEANS FAST LEARNING ARTIFICIAL NEURAL NETWORK

K-FLANN is an atomic module, suitable for creating scalable networks that mimic the behavior of biological neural systems.K-FLANN is used to cluster data efficiently and consistently with the minimal variation in cluster centroids and is not sensitive to the data presentation sequences (DPS) caused by randomly ordering the arrival of data[10].

2.1. Architecture

K-FLANN consists of two layers, input layer and output layer. The set of nodes in the input layer is determined based on the dimension of data. Input nodes are directly connected to output nodes by a set of weights. The number of nodes in the output layer grows as new groups formed in the clustering process. The dynamic formation of output nodes yield K-FLANN is a self-organized clustering ANN model. The architecture of the K-FLANN is shown in figure 1.

Figure 1. K-FLANN architecture



Algorithm:

Step 1 Initialize the network parameters.
.
Step 2 Present the pattern to the input layer. If there is no output node, GOTO step 6

Step 3 Determine all possible matches output node using Eq. (1).

$$\frac{\sum_{i=1}^{d} D\left[\delta_i^2 - (w_{ij} - x_i)^2\right]}{n} \geq \rho \qquad (1)$$

$$D[a] = \begin{cases} 1, & a > 0 \\ 0, & a \leq 0 \end{cases}$$

Step 4 Determine the winner from all matches output nodes using Eq. (2)

$$\text{Winner} = \min\left[\sum_{i=1}^{d} (w_{ij} - x_i)^2\right] \qquad (2)$$

Step 5 Match node is found. Assign the pattern to the match output node. Go to Step 2

Step 6 Create a new output node. Perform direct mapping of the input vector into weight vectors.

Step 7 After completing a single epoch, compute clusters
Centroid.If centroid points of all clusters unchanged, terminate
Else
GO TO Step 2.
Step 8 Find closest pattern to the centroid and re-shuffle it to the top of the dataset list, Go to Step 2.

Note: ρ is the Vigilance Value, δ_i is the tolerance for i^{th} feature of the input space, and W_{ij} is the cluster center of the corresponding output node denotes the weight connection of j_{th} output to the i_{th} input, Xi represent the i_{th} feature.

K-FLANN algorithm consists of two phases. Prior to clustering process, step 1 called initialization phase, is needed to determine vigilance and tolerance. The computation complexity depends on size of the data set and dimensionality of data set. The value of vigilance is usually between 0.5 to 1.Since the ρ value determines the ratio between the number of features to be within tolerance and the total number of features, the clustering formation becomes more stringent as the vigilance setting approaches 1.This parameter was adopted from [26]. The tolerance setting of features is the measurement of attributes dispersion, and thus computation is performed for every feature of the training data at the initial stage. Several methods were investigated to deal with tolerance setting [10]. The formula is given by the eq. (3).

$$\partial_i = \frac{\max diff + \min diff}{2} \tag{3}$$

Where

min diff – Minimum difference in attribute values for attribute i. This is the difference between the smallest and second smallest values of the attribute.

max diff – Maximum difference in attribute values for attribute i. This is the difference between the smallest and largest values of the attribute.

In the second phase, clustering process begins with the presentation of data patterns in sequence. Step 3 involves two types of computation called tolerance filtering and vigilance screening. Tolerance filtering shows local variation in the input features i.e. it determines whether the feature is within the acceptable tolerance or not.

It is implemented using discriminating function D[a] in step 3.The positive value of D[a] indicates that the particular attribute is within the bounds of the corresponding attribute of the cluster in question.

Vigilance screening shows global variation in the input features i.e. it determines the number of features within the localized input space that have passed tolerance tuning.

Competitive learning begins from step 4.After satisfying the screening criteria, there are still chances of exemplar mapped to more than one cluster. The final decision of ownership is decided by the WTA property of competitive learning. Eventually the cluster centroid bearing strongest weight will absorb the exemplar in to its cluster. The winning neuron is chosen based on the closest distance from the data point to centroid.

The exemplar is assigned as the new cluster member of the selected winner centriod in step 5.
In step 6, if the new exemplar does not meet the screening criteria, a new output node is created to represent new cluster.

After a single epoch, cluster stability is checked. New cluster centroids are calculated in step 7 and the data patterns closest to the centroids is determined.Re-shuffling the seed points closest to the cluster centroids to the top of the list creates new list of patterns for the next iterative clustering process in step 8.

2.2. Modifications in the K-FLANN Algorithm

The improvements in the K-FLANN algorithm are used to select the best matching unit for the formation of consistent clusters.

A step 4 of the K-FLANN algorithm is modified as follows:

Step 4 Determine the winner from all matched output nodes using the following criteria:

If same match is found

$$\text{Winner} = \min \left[\sum_{i=0}^{n} (w_{ij} - x_i)^2 \right] \tag{4}$$

Else

$$\text{Winner} = \max \left(\frac{\sum_{i=0}^{n} \left[\delta_i^2 - (w_{ij} - x_i)^2 \right]}{n} \right) \tag{5}$$

In the modified K-FLANN, a data point successfully satisfied the screening process and well suited as a member of more than one cluster with different match values to the centroids .The data point is assigned as a new cluster member of centroid if it has maximum matched value out of all matches. The winner centroid is the one with the highest matched value for a data point even if it is not closest to the data point. But in the original K-FLANN algorithm, a data point successfully satisfied the screening process and well suited as a member of more than one cluster, is assigned as a new cluster member of the selected winner centroid.The centroid closest to the data point is the Winner centroid because it has same match values for the other centroids.

3. DIFFERENTIAL EVOLUTION

The differential evolution (exploration) [DE] is a novel parallel direct search method, which utilizes NP parameter vectors as a population for each generation G.DE can be categorized into a class of floating-point encoded, evolutionary optimization algorithms. Currently, there are several variants of DE. The DE variant used throughout this investigation is the DE/rand/1/bin scheme [1][17].

The differential evolution (DE) algorithm is a population-based optimization method that simulates natural evolution combined with a mechanism to generate multiple search directions based on the distribution of solutions (vectors) in the current population. The DE algorithm's main strategy is to generate new individuals by calculating vector differences between other randomly selected individuals of the population.

Main Steps of the DE Algorithm

```
1) Definition of encoding scheme
2) Definition of selection criterion (fitness function)
3) Creation of population of chromosomes
4) Evaluation

Repeat
    Mutation
    Crossover
    Selection
Until (termination criteria are met).
```

In DE, a solution vector is randomly created at the start. This population is successfully improved by applying mutation, crossover and selection operators. Mutation and crossover are used to generate new vectors (trial vectors), and selection then are used to determine whether or not the new generated vectors can survive the next iteration [1]. The operators and equations are described in 3.1.

3.1. Definition of encoding scheme

Initial population is a set of NP ‘D’ dimensional parameter vectors known as genomes or chromosomes. Each solution vector consists of two components representing vigilance and tolerance parameters of KFLANN. Binary encoding is used for vigilance and floating point encoding is used for tolerance. Each parameter is of length ‘d’ in a solution vector.

For example, consider a data set consists of 5 features. The representation of a solution vector is shown below.

Bits of Vigilance parameter					Feature Tolerances(δ)					Vigilance(ρ)
1	0	1	1	1	0.2345	0.3156	0.2176	0.1023	0.0125	0.8

Fig. 2 structure of a solution vector

So the total length of a solution vector is ((2*D) +1) where ‘D’ is the number of features in a data set. For a feature, tolerance is the difference between maximum value and minimum value within the feature.

In the above example, if the feature to be matched, the corresponding bit is ‘1’ else the corresponding bit is ‘0’.So for four features the corresponding bit is ‘1’. So vigilance is 4/5=0.8.

3.2. Definition of fitness function

The main goal of clustering algorithms is maximizing the between group variance and minimizing within group variance. Any cluster validity measure can be used as fitness function. In this paper, a new cluster validity measure called CS measure proposed by Chou in 2004 is used as fitness function to evaluate the results of clustering and to select best vectors in the population. Also CS measure deals with clusters of different densities and/or sizes and is more efficient than other cluster validity indices.

The CS measure is a function of the ratio of the sums of within cluster scatter to between-cluster separation and similar to DB or DI indices. If the clusters are compact then within cluster scatter is minimum and between-cluster separation is maximum for well separated clusters. So the CS measure is defined as follows:

$$CS(K) = \frac{\frac{1}{K}\sum_{i=1}^{K}\left[\frac{1}{N_i}\sum_{\vec{X}_i \in C_i} \max_{\vec{X}_q \in C_i}\left\{d\left(\vec{X}_i, \vec{X}_q\right)\right\}\right]}{\frac{1}{K}\sum_{i=1}^{K}\left[\min_{j \in K, j \neq i}\left\{d\left(\vec{m}_i, \vec{m}_j\right)\right\}\right]} \quad (6)$$

$$= \frac{\sum_{i=1}^{K}\left[\frac{1}{N_i}\sum_{\vec{X}_i \in C_i} \max_{\vec{X}_q \in C_i}\left\{d\left(\vec{X}_i, \vec{X}_q\right)\right\}\right]}{\sum_{i=1}^{K}\left[\min_{j \in K, j \neq i}\left\{d\left(\vec{m}_i, \vec{m}_j\right)\right\}\right]} \quad (7)$$

Where

k- the number of clusters

$d\left(\vec{X}_i, \vec{X}_j\right)$ - is the distance metric between any two data points $\vec{x}_i$ and $\vec{x}_j$

$d\left(\vec{m}_i, \vec{m}_j\right)$ - is the distance metric between anv two centroids $\vec{m}_i$ and $\vec{m}_j$

The cluster centroid is calculated by taking the average of all data points that lies in a cluster. It is denoted by $\vec{m}_i$.

$$\vec{m}_i = \frac{1}{n} \sum_{x_j \in c_i} \vec{x}_j \tag{8}$$

The fitness function is the CS measure is combined with the mis-classification rate .The error rate is computed by comparing cluster structure produced by K-FLANN and externally supplied class labels. Error rate of a solution vector is the ratio of the number of samples misclassified and the total number of samples.

$$errorrate_i = \frac{mc}{total} x100 \tag{9}$$

Where *mc* is the no.of mis-classified samples
total is the total number of samples

Eq. (10) gives the fitness function of a solution vector ($\vec{X}_i$) based on the result of K-FLANN.

$$f\left(\overrightarrow{X_i}\right) = \frac{1}{CS_i(K) * errorrate_i + eps} \tag{10}$$

Where
CS_i (K) - CS measure calculated over K partitions
$errorrate_i$ - mis-classification rate of 'i'
eps - small bias term equal to 2x 10^{-6} and prevents the denominator of R.H.S from being equal to zero.

3.3. Mutation

To create $V_{i,G+1}$ for each ith member of the current population (also called the target vector), three other distinct parameter vectors, say the vectors $X_{r1,G}$, $X_{r2,G}$, and $X_{r3,G}$ are picked up randomly from the current population. For a solution vector, both binary mutation and floating point mutation is applied to create trial vector.

3.3.1. Binary mutation

In the binary mutation, logical OR operator is used to add a vector to the weighted difference between the two vectors. Difference of two vectors is done by Exclusive- OR operation. Multiplication is performed by logical AND operation.So mutation operation for binary vectors defined as follows:

For each binary target vector $X_{i,G}$ =1,2,3,. .., NP (population Size)

A binary mutant vector is produced by

$$V_{i,G+1} = X_{r1,G} \textbf{\textit{ OR }} F \textbf{\textit{ AND }} (X_{r2,G} \textbf{\textit{ XOR }} X_{r3,G}) \tag{11}$$

3.3.2. Floating- point mutation

DE generates new parameter vectors by adding the weighted difference between two population vectors to a third vector. This operation is called mutation. The mutated vector's parameters are then mixed with the parameters of another predetermined vector, the target vector, to yield the so-called trial vector.

For each target vector $X_{i,G} = 1,2,3,\ldots, NP$ (population Size).

A mutant vector is produced by

$$V_{i,G+1} = X_{r1,G} + F * (X_{r2,G} - X_{r3,G}) \tag{12}$$

Where $i, r1, r2, r3 \in \{1, 2, \ldots, NP\}$ are randomly chosen and must be different from each other.

F is the scaling factor, which controls the magnitude of the differential variation of $(x_{r2,G} - x_{r3,G})$.

NP is size of the population, which remains constant during the search process [1].

3.4. Crossover

The parent vector is combined with the mutated vector to produce a trial vector through a crossover operation which combines components from current element and mutant vector according to a control parameter CR

Two types of crossover schemes can be used for DE family of algorithms. In this work, binomial crossover is performed on each of the 'D' variables whenever a randomly picked number between 0 and 1.

```
If  ($rnd_j \le CR$)  or  $j = r_{ni}$
        $U_{ji,G+1} = U_{ji,G+1}$
Else
        $U_{ji,G+1} = X_{ji,G}$
End
```

Where

$j = 1, 2, \ldots, D$;

rnd_j random number generated in the interval [0 1];

CR is crossover constant $\in [0, 1]$;

$r_{ni} \in (1, 2, \ldots, D)$ is the randomly chosen index;

'D' represents the number of dimensions of a vector.

3.5. Selection

Selection is the process of deciding the better one by comparing the performance of trial vector and target vector. The newly formed trial vector is evaluated to decide whether it will survive to the next generation or not. If the trial vector produces a better fitness value i.e. lower value of objective function then it is copied to next generation otherwise target vector is passed into next generation [1].

The pseudo code for selection is as follows:

```
If $f(U_{i,G+1}) \leq f(X_{i,G})$
        $X_{i,G+1} = U_{i,G+1}$
Else
        $X_{i,G+1} = X_{i,G}$
End
```

4. K-FLANN AND DIFFERENTIAL EVOLUTION

The K-FLANN is competent clustering algorithm that provides consistent clusters with a short number of epochs. To partition a data set, K-FLANN needs optimal values of Vigilance and Tolerance. To initialize these parameters, a large search space is explored and best parameter values are determined using the differential evolution method. The encoding scheme is discussed in section 3.1. For each solution vector, K-FLANN runs and exits when stable centroids formed. The results of K-FLANN are evaluated using fitness function discussed in section 3.2.During a generation; new individuals are created using the differential evolution operators mutation and crossover discussed in section 3.3 and 3.4.The fitness values of trial vectors is compared with the fitness values of target vectors using selection explained in section 3.5.

5. EXPERIMENTAL RESULTS

5.1. Data sets used

The data sets used to carry this work are listed in Table 1.It gives detailed descriptions of both artificial data sets and synthetic data sets. The artificial data sets are downloaded from UCI machine learning repository [27].

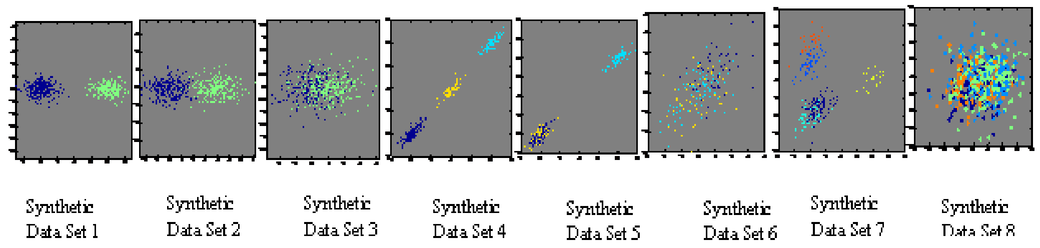
Table 1. Data sets description.

S.No.	Data Set	size	#of features	# of clusters
1	Iris	150	4	3(class 1-50, class 2-50, class 3-50)
2	Wine	178	13	3(class 1-59,class 2-70, class 3-49)
3	New Thyroid	215	5	3(class 1-150,class 2-35, class 3-30)
4	Haberman	306	3	2(class 1-227,class 2-79)
5	Glass	214	10	7(class1-70,class 2-76, class 3-17,class 4-0,class 5-13, class 6-9, class 7-29)
6	Synthetic data set 1	1000	2	2(class 1-500,class 2-500)
7	Synthetic data set 2	1000	2	2(class 1-500,class 2-500)
8	Synthetic data set 3	1000	2	2(class 1-500,class 2-500)
9	Synthetic data set 4	500	8	3(class 1-250, class 2-150, class 3-100)
10	Synthetic data set 5	400	8	3(class 1-150,class 2-150, class 3-100)
11	Synthetic data set 6	350	8	3(class 1-100,class 2-150, class 3-100)
12	Synthetic data set 7	450	12	5(class1-150, class 2-100,class3-100,class 4-50, class 5-50)
13	Synthetic data set 8	530	10	4(class 1-100,class 2-150,class 3-200, class 4-80)

5.2. Scatter plots for data sets

Figure 2 shows the scatter plots of 8 synthetic data sets.

Figure 2. Plots of synthetic data sets



5.3. Comparison of Modified K-FLANN and K-FLANN

The results of modified K-FLANN and K-FLANN were shown in two separate tables (table 2 and table 3) due to large size. The following section gives detailed information about them.

5.3.1. Results of DE applied to modified K-FLANN

The tables 2 and 3 shows average values of neural network parameters when DE runs about 100 runs and each run consists a maximum of 20 generations using both K-FLANN and Modified K-FLANN.

Table 2. Mean values of neural network parameters using DE and Modified K-FLANN

S.No	Data Set	Avg. no of clusters	Avg. error rate	Avg. fitness	Avg. tolerance	Avg. vigilance
1	iris	3	2.8067	0.6136	0.1442,0.4638,0.4834,0.1217	0.7050
2	Wine	3	5.8034	1.4750	0.2479,0.2717,0.3669,0.3685, 0.3747,0.2799,0.2845,0.3555, 0.2931,0.2444,0.2773,0.3239, 0.2593	0.5462
3	New Thyroid	3	4.9163	1.6056	0.2694,0.2091,0.3199,0.2804, 0.2109	0.5680
4	Glass	5.74	56.5327	0.7084	0.0134,0.0936,0.0589,0.0437, 0.0205,0.0809,0.1229,0.0435, 0.0185	0.7089
5	Haberman	2	25.2614	1.4151	0.1919,0.0572,0.1797	0.3333
6	Syndata1	2	0	0.4892	4.5914,0.8764	0.5
7	Syndata2	2	1.7090	0.8180	2.9595,1.0745	0.5
8	Syndata3	2	28.7020	1.6174	2.4482,1.0943	0.5050
9	Syndata4	3	0	0.3784	3.4852,3.3330,3.7854,3.5869, 3.2227,3.1327,3.1082,3.4594	0.82
10	Syndata5	3	0	0.2897	3.2933,4.1205,4.6169,5.1569, 2.6966,2.2702,4.7332,5.7182	0.8550
11	Syndata6	3.05(actual number is 3)	51.54	1.2910	1.9625,1.8903,1.9994,1.6769, 1.9692,2.2525,1.6881,1.8872	0.5963
12	Syndata7	4.46(actual number is 5)	23.9733	0.9146	3.7808,3.4297,3.3189,3.2467, 4.0519,3.5796,3.7147,3.1569, 3.8630,3.4178,4.3642,2.9537	0.6433
13	Syndata8	4.71(actual number is 4)	65.6019	1.8394	10.013,9.3882,10.753,11.976, 9.7804,11.717,11.075,9.6061, 8.3039,11.297	0.6490

5.3.2. Results of DE applied to K-FLANN

Table 3. Mean values of neural network parameters using DE and K-FLANN

S.No	Data set	Avg. no of clusters	Avg. error rate	Avg. fitness	Avg. tolerance	Avg.vigilance
1	Iris	3	3.3467	0.6234	0.1221,0.4657,0.5569,0.1145	0.74
2	Wine	3	6.7640	1.4553	0.3032,0.3196,0.3752,0.3378, 0.3195,0.2596,0.2985,0.3448, 0.3024,0.3413,0.3139,0.3429, 0.2899	0.6092
3	New thyroid	3	8.0279	1.3929	0.2953,0.2742,0.2539,0.2756 0.2019	0.61
4	Glass	5.88	53.1869	0.7151	0.0129,0.0841,0.0644,0.0405, 0.0217,0.0824,0.1306,0.0398, 0.0177	0.7011
5	Haberman	2	25.50	1.2946	0.1213,0.0492,0.3878	0.3333
6	Syndata1	2	0	0.4890	4.8529,0.8165	0.5
7	Syndata2	2	1.7970	0.8071	3.4698,0.8313	0.5
8	Syndata3	2	29.2	1.6555	2.3614,0.9826	0.5250
9	Syndata4	3	0	0.3784	3.4528,3.4220,3.6010,3.6698, 3.2752,3.1782,2.9961,3.2335	0.8187
10	Syndata5	3	0	0.2898	3.4314,3.2593,3.5499,3.5297, 3.6932,3.1813,3.3929,3.6039	
11	Syndata6	3.89(Actual number is 3)	53.3286	1.4002	1.8511,1.7823,2.0226,2.0770, 1.9317, 2.2599,2.0789,2.2137	0.6450
12	Syndata7	5	19.9444	0.9890	3.9368,3.8599,3.8655,3.0607, 3.5838,4.1074,3.8569,3.3853, 4.1354,3.5799,3.7644,2.9806	0.6908
13	Syndata8	4.75((Actual number is 4)	55.7340	1.8225	9.5979,9.0517,10.054,10.901, 10.954,11.417,9.6887,9.3184, 8.3506,9.8606	0.6240

The results also compare both the versions of K-FLANN combined with optimization algorithm called DE to determine best values of parameters vigilance and features tolerance. For most of the data sets used in this work, it was observed that modified K-FLANN is efficient than original K-FLANN in terms of error rate and fitness. Both algorithms were able to discover number of clusters actually present in the data set shown in the first column of the above two tables. The performance of modified K-FLANN is better than the original K-FLANN in terms of average error rate obtained over 100 runs. In each generation, the values of both tolerance and vigilance values are recorded and the mean value is taken for both parameters at the end which were shown in table 2 and table 3.

5.3.3. Results Based on maximum vigilance

Table 4 shows maximum vigilance and corresponding tolerance values obtained by DE for 100 runs.

Table 4. Comparison between modified K-FLANN and K-FLANN based on maximum vigilance

S.No	*Data set*	*Algorithm*	*Maximum vigilance(ρ)*	*Number of clusters*	*Features tolerance (δ)*	*Error rate (%)*
1	iris	Modified K-FLANN	0.75(3/4)	3	0.0429,0.6788,0.5608,0.1261	**2.6667**
		K-FLANN	0.759(3/4)	3	0.1818,0.3327,0.5247,0.1124	3.3333
2	Wine	Modified K-FLANN	0.6923(9/13)	3	0.4587,0.3101,0.2929,0.3157,0.3823, 0.3089,0.2822,0.4702,0.4058,0.3388, 0.3278,0.4672,0.2648	**5.0562**
		K-FLANN	0.6923(9/13)	3	0.2745,0.2005,0.2188,0.4898,0.4101, 0.4472,0.3386,0.3851,0.3528,0.3982, 0.3311,0.4017,0.4582	7.3034
3	New Thyroid	Modified K-FLANN	0.8(4/5)	3	0.4139,0.2167,0.4847,0.3654,0.4862	**4.6512**
		K-FLANN	0.8(4/5)	3	0.2568,0.4982,0.2404,0.3778,0.3267	8.8372
4	Glass	Modified K-FLANN	0.7778	6	0.01299,0.13299,0.0205, 0.0445,0.0189,0.0911, 0.1515,0.0191,0.0162	**50.4673**
		K-FLANN	0.8889	6	0.0178,0.0938,0.1048,0.0613,0.0269, 0.1193,0.2059,0.01981,0.01776	51.4019
5	Haberman	Modified K-FLANN	0.3333(1/3)	2	0.2955,0.0517,0.1892	**25.4902**
		K-FLANN	0.3333(1/3)	2	0.1405,0.0355,0.4385	**25.4902**
6	Syndata1	Modified K-FLANN	0.5(1/2)	2	4.8683,0.7714	**0**
		K-FLANN	0.5(1/2)	2	4.8529,0.8164	**0**
7	Syndata2	Modified K-FLANN	0.5(1/2)	2	3.1803,1.1039	**1.7**
		K-FLANN	0.5(1/2)	2	3.7621,0.6751	1.8
8	Syndata3	Modified K-FLANN	1(2/2)	2	3.1198,1.4604	**28.7**
		K-FLANN	1(2/2)	2	3.2106,1.4721	29.2
9	Syndata4	Modified K-FLANN	1(8/8)	3	5.4434,5.3906,5.0819,4.3982,3.1581, 3.1956,4.9923,4.4596	**0**
		K-FLANN	1(8/8)	3	5.0968,2.3999,3.7814,5.1546,5.4707, 5.0594, 4.1309,4.7899	**0**
10	Syndata5	Modified K-FLANN	1(8/8)	3	3.2933,4.1205,4.6169,5.1569,2.6966, 2.2702,4.7333,5.7182	**0**
		K-FLANN	1(8/8)	3	4.5044,2.8447,5.1821,4.7824,5.6415, 2.2199,5.3356,5.1545	**0**
11	Syndata6	Modified K-FLANN	0.75(6/8)	4(Actual number is 3)	2.0628,2.3158,2.6881,0.8054,1.9819, 3.0941,1.7463,1.3778	55.1429
		K-FLANN	0.75(6/8)	4 (Actual number is 3)	2.0256,1.4159,2.4478,2.8806,1.3764, 3.2083,1.0430,2.7278	**52.5714**
12	Syndata7	Modified K-FLANN	0.8333(10/12)	4(Actual number is 5)	3.2019,3.4565,3.9432,4.0429,5.9774, 3.3968,4.3376,0.4221,6.3798,5.6094, 4.9701,3.9043	26.2222
		K-FLANN	0.8333(10/12)	5	5.8289,6.2038,3.4297,3.1956,5.0315, 4.1359,3.7991,2.8922,6.3156,4.8360, 5.0133,0.5451	**19.7778**
13	Syndata8	Modified K-FLANN	0.8(8/10)	5(actual number is 4)	11.0751,14.2607,8.6679,12.899,6.466 3,6.855,9.6163,13.619,13.702,14.202	65.8491
		K-FLANN	0.8(8/10)	6(actual number is 4)	10.5066,16.1200,10.7783,18.4743, 6.71937,9.04243,13.3576,7.00378, 11.1675,10.5216	**47.5472**

The error rate was shown for maximum vigilance because vigilance tells about the number of features that are within their tolerance out of total features lies in the data set. The values in bold shows minimum error rate obtained by the two variations of K-FLANN.It was observed that modified K-FLANN performs well for iris, wine, new thyroid, Glass, etc.It was shown that, the performance of both algorithms i.e. the error rate is same for all well separated data sets (syndata1, syndata2, syndata 4 and syndata5). For overlapped data sets syndata 7 and syndata 8 K-FLANN is efficient due to low error rate for maximum vigilance observed over all 100 runs. For most of the data sets the maximum vigilance value is same with different feature tolerances and mis-classification rate.

5.3.4. Results Based On Minimum Error Rate

Table 5 shows the minimum error rate obtained for each data set over all 100 runs. For each data set, the number of clusters formed, tolerances of each feature and vigilance value is recorded for minimum error rate. For iris data set, the vigilance parameter value is same for both the algorithms with different tolerance values. But mis-classification rate is minimized when modified K-FLANN is used. In the case of wine, vigilance value is small (0.4615 i.e. 6 features matched out of 13) with low mis-classification rate. But using K-FLANN, the vigilance value is 0.6923 i.e. 9 features matched out of 13, but error rate is high. In the similar way the analysis can be done for the other data sets also. For a data set, the obtained feature tolerances and vigilance values can be used for the K-FLANN algorithms to determine groups in a data set.

Table 5. Comparison between modified K-FLANN and K-FLANN based on minimum error rate

S.No	*Data set*	*Algorithm*	*Minimum error rate*	*Number of clusters*	*Tolerances of features(δ)*	*Vigilance(ρ)*
1	iris	Modified K-FLANN	**2.6667**	3	0.0429,0.6788,0.5608,0.1262	0.75(3/4)
		K-FLANN	3.3333	3	0.1818,0.3327,0.5247,0.1124	0.75(3/4)
2	Wine	Modified K-FLANN	**3.3708**	3	0.2309,0.1097,0.4529,0.2372,0.3139, 0.1649,0.4125,0.2753,0.4576,0.0564, 0.1988,0.1671,0.4866	0.4615(6/13)
		K-FLANN	3.9326	3	0.2799,0.2539,0.4122,0.3773,0.1221, 0.4154,0.2669,0.24990.3865,0.4822, 0.4918,0.4863,0.4339	0.6923(9/13)
3	New Thyroid	Modified K-FLANN	**3.7209**	3	0.3133,0.1179,0.2027,0.3079,0.3809	0.6(3/5)
		K-FLANN	5.1163	3	0.1305,0.1034,0.0917,0.4774,0.0383	0.4(2/5)
4	Glass	Modified K-FLANN	49.0654	6	0.0181,0.1098,0.0986,0.0245,0.0259, 0.1041,0.1064,0.0141,0.0175	0.7778
		K-FLANN	**45.7944**	6	0.0181,0.0963,0.0418,0.0426,0.0229, 0.0953, 0.2232,0.0534,0.0219	0.7778
5	Haberman	Modified K-FLANN	**23.2026**	2	0.0744,0.0580,0.1927	0.3333(1/3)
		K-FLANN	25.1634	2	0.2032,0.0384,0.1752	0.3333(1/3)
6	Syndata 1	Modified K-FLANN	0	2	4.8682,0.7713	0.5(1/2)
		K-FLANN	0	2	5.1367,0.6858	0.5(1/2)
7	Syndata 2	Modified K-FLANN	**1.6**	2	2.8330,0.2483	0.5(1/2)
		K-FLANN	1.7	2	3.4380,0.8119	0.5(1/2)
8	Syndata 3	Modified K-FLANN	**28.7**	2	1.9122,1.2469	0.5(1/2)
		K-FLANN	29.2	2	2.5386,0.8335	0.5(1/2)

9	Syndata 4	Modified K-FLANN	0	3	3.8012,1.9051,2.8838,3.0330,2.1734, 2.9809,3.3110,2.5897	0.75(6/8)
		K-FLANN	0	3	3.8026,4.3033,5.2903,4.1085,3.5378, 4.7799, 0.2037,3.3766	0.75(6/8)
10	Syndata 5	Modified K-FLANN	0	3	1.4741,3.9279,3.5081,4.9405,4.1747, 4.6311,2.6078,2.4269	0.8750(7/8)
		K-FLANN	0	3	2.0981,2.1447,3.3083,4.4443,4.8721, 2.6208,1.2238,5.6164	0.8750(7/8)
11	Syndata 6	Modified K-FLANN	**47.1429**	3	0.4076,0.7234,2.7625,1.2517,2.5615, 2.2389,1.8712, 2.2224	0.6250(5/8)
		K-FLANN	48.2857	5(Actual number is 3)	2.5104,2.0907,1.8238,2.5572,2.3843, 3.0045,1.5282,2.0187	0.6250(5/8)
12	Syndata 7	Modified K-FLANN	**15.7778**	5	1.1336,4.6821,2.8038,2.5385,4.5301, 1.5936,3.9037,4.3959,2.4807,4.0601, 5.3151,3.5744	0.6667(8/12)
		K-FLANN	19.7778	5	3.4639,2.5332,6.3620,4.7228,4.4233, 0.9349,2.0356,3.9908,6.1404,2.2257, 4.8480,3.9685	0.6667(8/12)
13	Syndata 8	Modified K-FLANN	49.8113	4	9.2010,6.0452,10.7022,16.7049, 16.2466,9.6262,8.6508,14.0265, 9.4569,17.0305	0.7(7/10)
		K-FLANN	**40**	4	12.8362,11.4845,13.7928,16.7031, 14.8295,11.2415,3.5456,11.7959, 10.4237,8.7383	0.7(7/10)

6. CONCLUSIONS

In this work, the optimization algorithm is applied to neural network to set initial parameters called vigilance (ρ) and tolerance (δ). From the above results, it was observed that the number of output nodes generated (centroids) depends on these parameters. The nodes created in output layer give the number of clusters present in a data set. So using efficient values of the parameters found using DE, optimal clustering is achieved. Also original K-FLANN algorithm is modified to determine the best matching unit out of the total matched nodes to improve accuracy. The above results show that modified K-FLANN is efficient to discover actual number of clusters presents in a data set with the best values of vigilance and tolerance lies in a solution vector. In this paper, the comparison is also made between original K-FLANN and modified K-FLANN.The above results show that the percentage of tuples mis-classified (error rate) is minimum using modified K-FLANN compared to original K-FLANN. In some cases, the original K-FLANN is proficient than modified K-FLANN. So in future, modified K-FLANN can be improved for highly overlapped data sets. In future, the algorithm can be extended to apply for medical imaging applications.

REFERENCES

[1] R. Storn, K. Price (1997), “Differential Evolution: A Simple and Efficient Heuristic for global Optimization over Continuous Spaces,” Journal of Global Optimization, vol. 11, no. 4, pp. 341–359.

[2] Holland JH (1975), Adaptation in Natural and Artificial Systems, University of Michigan Press, Ann Arbor.

[3] Goldberg DE (1975), Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley, Reading, MA.

[4] Brian Hegerty, Chih-Cheng Hung, and Kristen Kasprak (2009),” A Comparative Study on Differential Evolution and Genetic Algorithms for Some Combinatorial Problems”, Website: http://www.micai.org/2009/proceedings/.../cd/ws.../paper88.micai09.pdf

[5] Xin-SheYang (2011), Metaheuristic optimization,Scholarpedia,doi:10.4249/ scholarpedia.11472

[6] Carpenter, G. A. and Grossberg, S. (1987). “A Massively Parallel Architecture for a Self-Organizing Neural Pattern Recognition Machine”, Computer Vision, Graphics, and Image Processing, Vol. 37, pp.54-115.

[7] L.P.Tay, D.J. Evans (1994), “Fast Learning Artificial Neural Network Using Nearest Neighbors Recall”, Neural Parallel and Scientific Computations, 2(1).

[8] D.J.Evans, L.P.Tay (1995), “Fast Learning Artificial Neural Network for continuous Input Applications”, Kybernetes, 24 (3).

[9] Alex T.L.P., Sandeep Prakash (2002), “K-Means Fast Learning Artificial Neural Network, An Alternate Network for Classification”, ICONIP’02, Vol. 2, pp. 925 - 929.

[10] L. P. Wong, T. L. P. Alex (2003), “Centroid Stability with K-Means Fast Learning Artificial Neural Networks”, International Joint Conference on Neural Network (IJCNN).

[11] J, Vesterstrom, R. Thomsen (2004), “A comparative study of differential evolution, particle swarm optimization and evolutionary algorithms on numerical benchmark problems”. Congress on Evolutionary Computation, pp. 980-987.

[12] U. Chakraborty (2008), Advances in Differential Evolution (Studies in Computational Intelligence, vol. 142), Heidelberg, Germany: Springer-Verlag.

[13] V. Feoktistov (2006), Differential Evolution—In Search of Solutions (Springer Optimization and Its applications, vol. 5). Heidelberg, Germany: Springer-Verlag.

[14] K. V. Price, R. M. Storn, and J. A. Lampinen, Differential Evolution—A Practical Approach to global Optimization (Natural Computing Series).Heidelberg, Germany: Springer-Verlag, 2005.

[15] J. Zhang and A. C. Sanderson, “JADE: Adaptive differential evolution with optional external archive,” IEEE Transactions on Evolutionary Computing. vol. 13, no. 5, pp. 945–958, Oct. 2009.

[16] Daniela Zaharie (2009), “Influence of crossover on the behavior of Differential Evolution Algorithms”, Applied Soft Computing, volume 9, issue 3, pp.1126-1138

[17] Godfrey Onwubolu, Donald Davendra (2006), “Discrete Optimization Scheduling flow shops using differential evolution algorithm”, European Journal of Operational Research, vol.171, pp.674–692.

[18] http://www.maths.uq.edu.au/MASCOS/Multi-Agent04/Fleetwood.pdf

[19] Price K.V. (1999), ‘An Introduction to Differential Evolution’ in Corne, D., Dorigo, M. and Glover, F. (eds), New Ideas in Optimization, McGraw-Hill, London.

[20] C.H. Chou, M.C. Su, E. Lai (2004),” A new cluster validity measure and its application to image compression”, Pattern Analysis Applications, vol.7, pp.205–220, DOI 10.1007/s10044-004-0218-1

[21] Bandyopadhyay S., and Maulik U. (2002) "An evolutionary technique based on K-means algorithm for optimal clustering in RN", Information Sciences, vol. 146, pp.221-237.

[22] Storn R., and Kenneth P. (1995), "Differential Evolution – a Simple and Efficient Adaptive Scheme for Global Optimization over Continuous Spaces". Technical Report, TR-95.012, ICSI.

[23] Maulik U. and Bandyopadhyay S. (2000), "Genetic algorithm based clustering technique", Pattern Recognition, Vol. 33, pp.1455-1465.

[24] Yin Xiang, Alex Tay Leng Phuan (2004) ,”Genetic algorithm based k-means fast learning artificial neural network”.AI’04 proceedings of the 17th Australian joint conference on Advances in Artificial Intelligence, pp:828-839 (2004), ISBN: 3-540-24059-4 978-3-540-24059-4.

[25] Swagatam Das, Ajith Abraham and Amit Konar (2009),”Metaheuristic Clustering” (studies in computational intelligence vol.178), Heidelberg, Berlin: springer-verlag.

[26] S.Grossberg (1978), “Competitive Learning: from Interactive activation to adaptive resonance”, Cognitive science, 11, pp.23-96.

[27] Blake, C.L. and C.J. Merz, 1998. UCI Repository of Machine Learning Databases. 1st Edn, University of California, Irvine, CA.