European Journal of Academic Essays, Special Issue (1): 9-17, 2014
ISSN: 2183-1904
www.euroessays.org

Recent Trends in Analysis of Algorithms and Complexity Theory

SFLANN for Classification of Life Science Databases

<Satapathy, Lokanath>[1], <Misra, Bijan B.>[2], <Biswal, Bhabendra N.>[3]

[1]School of Computer Application,
KIIT University,
Koel Campus, Patia,
Bhubaneswar-751024, India
lokanathsatapathy@gmail.com

[2]Department of Information Technology,
Silicon Institute of Technology,
Bhubaneswar-751024, India
misrabijan@gmail.com

[3]Bhubaneswar Engineering College,
Paniora, NK Nagar, Pittapally
Bhubaneswar - 751054,
Odisha, INDIA
bhabendra_biswal@yahoo.co.in

Abstract: This paper presents SFLANN, a hybrid model of Particle Swarm Optimization (PSO) and Functional Link Artificial Neural Network (FLANN) for classification of life science databases. Backpropagation neural networks (BPN) are most commonly used technique for data classification. BPN usually suffers from many drawbacks, such as how many layers to be taken, how many neurons in each hidden layer, what activation function to be used, how to overcome local traps etc. FLANN is a single layer network and to avoid the drawbacks of single layer network, the input values are functionally expanded to map complex non-linear problems. Further to get rid of the local traps in backpropagation, here PSO technique is used to optimize the weights. Four different standard classifiers BPN, naïve Bayes classifier, k-nearest neighbor, multiple linear regression (MLR) classifiers are compared to evaluate the performance of SFLANN. 6 bench marked life science databases are considered to evaluate the performance of these models. Simulation results show that the classification accuracy obtained by SFLANN is better in comparison to other classifier models at most of the instances.

Keywords: Particle Swarm Optimization, Functional Link Artificial Neural Network, Backpropagation neural network, Classification, K-nearest Neighbor, Multiple Linear Regression**.**

---

1. Introduction

For the past few years, there have been many studies [1] focused on the classification task in the emerging field of data mining. In classification, we are given a set of example records, called a training set, where each record consists of several fields or attributes. Attributes are either continuous, coming from an ordered domain or categorical coming from an unordered domain. One of the attributes called the classifying attribute indicates the class to which each example belongs. The objective of classification is to build a model of the classifying attribute based upon the other attributes.

Several classification models have been proposed over the years, e.g. statistical models like linear/quadratic discriminates [2], genetic models [3], decision trees [4,5] and neural networks [6]. Among these we found very rare literatures on neural networks specifically FLANN for classification task of data mining. It is sometimes difficult to search the optimal nonlinear boundary for a classification problem, other than neural network models. Hence the nonlinear learning capabilities of artificial neural networks (ANNs) have become a powerful tool for many complex applications including functional approximation, nonlinear system identification and control, unsupervised classification, and optimization. The ANN's are capable of generating complex mapping between the input and the output space and thus these networks can form arbitrarily complex nonlinear decision boundaries. The traditional algorithms also take longer time to optimize the weight vectors and their complexity increases as the number of layers increases. Hence, to resolve few of the issues, in this paper we use functional link artificial neural network for solving the

Corresponding Author: Lokanath Satapathy

classification problem and to avoid local traps the weights are optimized by PSO.

Pao et al. [7] was originally proposed the FLANN architecture. They have shown that, their proposed network may be conveniently used for function approximation and pattern classification with faster convergence rate and lesser computational load than an MLP structure. The FLANN is basically a flat net and the need of the hidden layer is removed and hence, the learning algorithm used in this network becomes very simple. The functional expansion effectively increases the dimensionality of the input vector and hence the hyper planes generated by the FLANN provide greater discrimination capability in the input pattern space.

The rest of this paper is organized as follows. In Section 2, we have discussed basics of FLANN architecture. Section 3 illustrates the PSO technique. Section 4 provides FLANN for Classification. In Section 5, SFLANN architecture is presented. Simulation results are analyzed in Section 6. Section 7 concludes the article.

2. BASICS OF FLANN ARCHITECTURE

2.1 Introduction to Artificial Neural Networks

Over the past decade, Artificial Neural Network (ANN) has become increasingly popular in many disciplines as a problem solving tool. ANN has the ability to solve extremely complex problems with highly non-linear relationships. ANN's flexible structure is capable of approximating almost any input output relationships. Particularly ANN has been extensively used as a tool in many disciplines to solve different types of problems such as *forecasting*, *identification* and *control*, *classification*, and *optimization*. Complex and heterogeneous systems are extremely difficult to model mathematically. However, it has been proved that ANN's flexible structure can provide simple and reasonable solutions to various problems.

2.2 A Formal Computational Model of Neural Network

Let us first recall a general model of an artificial neural network that consists of *s* simple computational *units* or *neurons*, indexed as $V = \{1, \ldots, s\}$, where $s = |V|$ is called the network *size*. Some of these units may serve as external inputs or outputs, and hence we assume that the network has *n input* and *m output* neurons, respectively. The remaining ones are called *hidden* neurons. The units are densely connected into an oriented graph representing the *architecture* of the network, in which each edge *(i, j)* leading from neuron *i* to *j* is labeled with a real *(synaptic) weight* $w(i, j) = w_{ji} \in \Re$ The absence of a connection within the architecture corresponds to a zero weight between the respective neurons.

The *computational dynamics* of a neural network determines for each neuron $j \in V$ the evolution of its real *state* (*output*) $y_j^{(t)} \in \Re$ as a function of time $t \geq 0$. This establishes the *network state* $y^{(t)} = \left(y_1^{(t)}, \ldots, y_s^{(t)}\right) \in \Re^s$ at each time instant $t \geq 0$. At the beginning of a computation, the neural network is placed in an *initial state* $y^{(0)}$, which may also include an external input. Typically, a network state is updated by a selected subset of neurons collecting their *inputs* from the outputs of their incident neurons via the underlying weighted connections and transforming these input values into their current states. Finally, a global output from the network is read at the end of computation, or even in the course of it.

In general the models that we use to solve complex problems are multi-layer neural network. There are many algorithms to train the neural network models. However the models being complex in nature, one single algorithm cannot be claimed as best for training to suit different scenarios of the complexities of real life problems. Depending on the complexities of the problems, the number of layer and number of neuron in the hidden layer need to be changed. As the number of layers and the number of neurons in the hidden layer increases, training the model becomes further complex. Very often different algorithms fail to train the model for a given problem set. However we try to find an alternative algorithm, which will train the model to provide us with an output possibly not good enough to our expectation. In the process we develop one model containing many hidden layers and neurons, which is very complex to train, and computation intensive.

2.3 FLANN Architecture

To overcome the complexities associated with multi-layer neural network, single layer neural network can be considered as an alternative approach. But the single layer neural network being linear in nature very often fails to map the complex nonlinear problems. The classification task in data mining is highly nonlinear in nature. So solving such problems in single layer feed forward artificial neural network is almost an impossible task.

To bridge the gap between the linearity in the single layer neural network and the highly complex and computation intensive multi layer neural network, the FLANN architecture is suggested [1]. The FLANN architecture uses a single layer feed forward neural network and to overcome the linear mapping, functionally expands the input vector.

Let each element of the input pattern before expansion be represented as $z(i), 1 < i < d$ where each element z(i) is functionally expanded as $z_n(i), 1 < n < N$, where N = number of expanded points for each input element. Expansion of each input pattern is done as in (1).

$$\left.\begin{array}{l} z_1(i) = x_i, \\ z_2(i) = f_2(x_i), \\ \ldots, \\ z_N(i) = f_N(x_i) \end{array}\right\} d, \text{ d is the set of features in the dataset.} \quad (1)$$

These expanded input pattern are then fed to the single layer neural network and the network is trained to obtain the desired output. The set of functions considered for function expansion may not be always suitable for mapping the nonlinearity of the complex task. In such cases few more

functions may be incorporated to the set of functions considered for expansion of the input dataset. However dimensionality of many problems itself are very high and further increasing the dimensionality by to a very large extent may not be an appropriate choice. So, it is advisable to choose a small set of alternate functions, which can map the function to the desired extent.

3. Particle Swarm Optimization

3.1 PSO basics

The implicit rules adhered to by the members of bird flocks and fish schools, that enable them to move synchronized, without colliding, resulting in an amazing choreography, was studied and simulated by several scientists [8, 9]. In simulations, the movement of the flock was an outcome of the individuals' (birds, fishes etc.) efforts to maintain an optimum distance from their neighboring individuals [8].

The social behavior of animals, and in some cases of humans, is governed by similar rules [11]. However, human social behavior is more complex than a flock's movement. Besides physical motion, humans adjust their beliefs, moving, thus, in a belief space. Although two persons cannot occupy the same space of their physical environment, they can have the same beliefs, occupying the same position in the belief space, without collision. This abstractness in human social behavior is intriguing and has constituted the motivation for developing simulations of it. There is a general belief, and numerous examples coming from nature enforce the view, that social sharing of information among the individuals of a population may provide an evolutionary advantage. This was the core idea behind the development of PSO [10].

3.2 PSO Algorithm

PSO's precursor was a simulator of social behavior that was used to visualize the movement of a birds' flock. Several versions of the simulation model were developed, incorporating concepts such as nearest-neighbor velocity matching and acceleration by distance [10, 12]. When it was realized that the simulation could be used as an optimizer, several parameters were omitted, through a trial and error process, resulting in the first simple version of PSO [10].

PSO is similar to EC techniques in that, a population of potential solutions to the problem under consideration is used to probe the search space. However, in PSO, each individual of the population has an *adaptable velocity* (position change), according to which it moves in the search space. Moreover, each individual has a *memory*, remembering the best position of the search space it has ever visited [10]. Thus, its movement is an aggregated acceleration towards its best previously visited position and towards the best individual of a topological neighborhood. Since the "acceleration" term was mainly used for particle systems in Particle Physics [13], the pioneers of this technique decided to use the term *particle* for each individual, and the name *swarm* for the population, thus, coming up with the name *Particle Swarm* for their algorithm [12].

Two variants of the PSO algorithm were developed, one with a global neighborhood, and the other with a local neighborhood. According to the global variant, each particle moves towards its best previous position and towards the best particle in the whole swarm. On the other hand, according to the local variant, each particle moves towards its best previous position and towards the best particle in its restricted neighborhood [10]. In the following paragraphs, the global variant is exposed.

Suppose that the search space is *D*-dimensional, then the i^{th} particle of the swarm can be represented by a *D*-dimensional vector, $x_i = (x_{i1}, x_{i2}, ..., x_{iD})$. The *velocity* (position change) of this particle, can be represented by another *D*-dimensional vector $v_i = (v_{i1}, v_{i2}, ..., v_{iD})$. The best previously visited position of the i^{th} particle is denoted as $p_i = (p_{i1}, p_{i2}, ..., p_{iD})$. Defining *g* as the index of the best particle in the swarm (i.e., the g^{th} particle is the best), and let the superscripts denote the iteration number, then the swarm is manipulated according to the following two equations:

$$v_{id}^{n+1} = v_{id}^{n} + cr_1^n\left(p_{id}^{n} - x_{id}^{n}\right) + cr_2^n\left(p_{gd}^{n} - x_{id}^{n}\right) \quad (2)$$

$$x_{id}^{n+1} = x_{id}^{n} + v_{id}^{n+1} \quad (3)$$

where $d = 1, 2, \ldots, D$; $i = 1, 2, \ldots, N$, and N is the size of the swarm; c is a positive constant, called *acceleration constant*; $r1$, $r2$ are random numbers, uniformly distributed in [0, 1]; and $n = 1, 2, \ldots$, determines the iteration number.

Equations (2) and (3) define the initial version of the PSO algorithm. Since there was no actual mechanism for controlling the velocity of a particle, it was necessary to impose a maximum value *Vmax* on it. If the velocity exceeded this threshold, it was set equal to *Vmax*. This parameter proved to be crucial, because large values could result in particles moving past good solutions, while small values could result in insufficient exploration of the search space. This lack of a control mechanism for the velocity resulted in low efficiency for PSO, compared to EC techniques [14]. Specifically, PSO located the area of the optimum faster than EC techniques, but once in the region of the optimum, it could not adjust its velocity step size to continue the search at a finer grain.

The aforementioned problem was addressed by incorporating a weight parameter for the previous velocity of the particle. Thus, in the latest versions of the PSO, Equations (2) and (3) are changed to the following ones [15, 16, 17]:

$$v_{id}^{n+1} = \chi\left(wv_{id}^{n} + c_1r_1^n\left(p_{id}^{n} - x_{id}^{n}\right) + c_2r_2^n\left(p_{gd}^{n} - x_{id}^{n}\right)\right) \quad (4)$$

$$x_{id}^{n+1} = x_{id}^{n} + v_{id}^{n+1} \quad (5)$$

where w is called *inertia weight*; $c1$, $c2$ are two positive constants, called *cognitive* and *social* parameter respectively; and χ is a *constriction factor*, which is used, alternatively to w to limit velocity.

In the local variant of PSO, each particle moves towards the best particle of its neighborhood. For example, if the size of the neighborhood is 2, then the i^{th} particle moves towards the best particle among the $(i-1)^{th}$, the $(i+1)^{th}$ and itself.

The PSO method appears to adhere to the five basic principles of swarm intelligence, as defined by [10]:

(a) *Proximity*, i.e., the swarm must be able to perform simple space and time computations;

(b) *Quality*, i.e., the swarm should be able to respond to quality factors in the environment;

(c) *Diverse response*, i.e., the swarm should not commit its activities along excessively narrow channels;

(d) *Stability*, i.e., the swarm should not change its behavior every time the environment alters; and finally
(e) *Adaptability*, i.e., the swarm must be able to change its behavior, when the computational cost is not prohibitive.

Indeed, the swarm in PSO performs space calculations for several time steps. It responds to the quality factors implied by each particle's best position and the best particle in the swarm, allocating the responses in a way that ensures diversity. Moreover, the swarm alters its behavior (state) only when the best particle in the swarm (or in the neighborhood, in the local variant of PSO) changes, thus, it is both adaptive and stable [10].

3.3The parameters of PSO

The role of the *inertia weight w*, in Equation (4), is considered critical for the PSO's convergence behavior. The inertia weight is employed to control the impact of the previous history of velocities on the current one. Accordingly, the parameter *w* regulates the trade-off between the global (wide-ranging) and local (nearby) exploration abilities of the swarm. A large inertia weight facilitates global exploration (searching new areas), while a small one tends to facilitate local exploration, i.e., fine-tuning the current search area. A suitable value for the inertia weight *w* usually provides balance between global and local exploration abilities and consequently results in a reduction of the number of iterations required to locate the optimum solution. Initially, the inertia weight was constant. However, experimental results indicated that it is better to initially set the inertia to a large value, in order to promote global exploration of the search space, and gradually decrease it to get more refined solutions [16, 17]. Thus, an initial value around 1.2 and a gradual decline towards 0 can be considered as a good choice for *w*.

The parameters $c1$ and $c2$, in Equation (4), are not critical for PSO's convergence. However, proper fine-tuning may result in faster convergence and alleviation of local minima. An extended study of the acceleration parameter in the first version of PSO, is given in [18]. As default values, $c1 = c2 = 2$ were proposed, but experimental results indicate that $c1 = c2 = 0.5$ might provide even better results. Recent work reports that it might be even better to choose a larger cognitive parameter, $c1$, than a social parameter, $c2$, but with $c1 + c2 <= 4$ [19].
The parameters $r1$ and $r2$ are used to maintain the diversity of the population, and they are uniformly distributed in the range [0, 1]. The constriction factor χ controls on the magnitude of the velocities, in a way similar to the *Vmax* parameter, resulting in a variant of PSO, different from the one with the inertia weight.

4. FLANN FOR CLASSIFICATION

4.1 Classification

The digital revolution has made digitized information easy to capture and fairly inexpensive to store [20, 21]. With the development of computer hardware and software and the rapid computerization of business, huge amount of data have been collected and stored in databases. The rate at which such data stored is growing at a phenomenal rate. As a result, traditional ad-hoc mixtures of statistical techniques and data management tools are no longer adequate for analyzing this vast collection of data.

Raw data is rarely of direct benefit. Its true value is predicated on the ability to extract information useful for decision support or exploration, and understanding the phenomenon governing the data source. In most domains, data analysis was traditionally a manual process. One or more analysts would become intimately familiar with the data and, with the help of statistical techniques, provide summaries and generate reports. In effect, the analyst acted as a sophisticated query processor. However, such an approach rapidly breaks down as the size of data grows and the number of dimensions increases. When the scale of data manipulation, exploration and inferencing goes beyond human capacities, people look to computing technologies for automating the process.

All these have prompted the need for intelligent data analysis methodologies, which could discover useful knowledge from data. The term KDD refers to the overall process of knowledge discovery in databases. Data mining is a particular step in this process, involving the application of specific algorithms for extracting patterns (models) from data [22]. Supervised pattern classification is one of the important tasks of data mining.

Supervised pattern classification can be viewed as a problem of generating appropriate class boundaries, which can successfully distinguish the various classes in the feature space [23]. In real-life problems, the boundaries between different classes are usually nonlinear. It is known that any nonlinear surface can be approximated by using a number of hyperplanes. Hence, the problem of classification can be viewed as searching for a number of linear surfaces that can appropriately model the class boundaries while providing minimum number of misclassified data points.

The goal of pattern classification [24] is to assign input patterns to one of a finite number, *M*, of classes. In the following, it will be assumed that input patterns consist of static input vectors $\boldsymbol{x}$ containing $\boldsymbol{N}$ elements or continuous valued real numbers denoted $x_1, x_2, ..., \boldsymbol{x_N}$. Elements represent measurements of features selected to be useful for distinguishing between classes. Input patterns can be viewed as points in the multidimensional space defined by the input feature measurements. The purpose of a pattern classifier is to partition this multidimensional, space into decision regions that indicate to which class any input belongs. Conventional Bayesian classifiers characterize classes by their probability density functions on the input features and use Bayes' decision theory to form decision regions from these densities [25, 26]. Adaptive non-parametric classifiers do not estimate probability density functions directly but use discriminant functions to form decision regions.

Application of a pattern classifier first requires selection of features that must be tailored separately for each problem domain. Features should contain information required to distinguish between classes, be insensitive to irrelevant variability in the input, and also be limited in number to permit efficient computation of discriminant functions and to limit the amount of training data required. Good

classification performance requires selection of effective features and also selection of a classifier that can make good use of those features with limited training data, memory, and computing power. Following feature selection, classifier development requires collection of training and test data, and separate training and test or use phases. During the training phase, a limited amount of training data and a ***priori*** knowledge concerning the problem domain is used to adjust parameters and/or learn the structure of the classifier. During the test phase, the classifier designed from the training phase is evaluated on new test data by providing a classification decision for each input pattern. Classifier parameters and/or structure may then be adapted to take advantage of new training data or to compensate for nonstationary inputs, variation in internal components, or internal faults. Further evaluations require new test data.

It is important to note that test data should never be used to estimate classifier parameters or to determine classifier structure. This will produce an overly optimistic estimate of the real error rate. Test data must be independent data that is only used to assess the generalization of a classifier, defined as the error rate on never-before-seen input patterns. One or more uses of test data, to select the best performing classifier or the appropriate structure of one type of classifier, invalidate the use of that data to measure generalization. In addition, input features must be extracted automatically without hand alignment, segmentation, or registration. Errors caused by these processes must be allowed to affect input parameters as they would in practical applications where extensive hand-tuning is normally impossible. Unfortunately, these simple guidelines, restricting use of test data and limiting hand-tuning, and also other important common-sense guidelines discussed in [27, 28], are frequently broken by pattern recognition researchers.

Supervised training, unsupervised training, or combined unsupervised/supervised training can be used to train neural net classification and clustering algorithms. Classifiers trained with supervision require data with side information or labels that specify the correct class during training. Clustering or vector quantization algorithms use unsupervised training and group unlabeled training data into internal clusters. Classifiers that use combined unsupervised/supervised training typically first use unsupervised training with unlabeled data to form internal clusters. Labels are then assigned to clusters and cluster centroid locations, and sizes are often altered using a small amount of supervised training data. Although combined unsupervised/supervised training mimics some aspects of biological learning, it is of interest primarily because it can reduce the amount of labeled training data required. Much of the expense and effort required to develop classifiers results from the necessity of collecting and hand-labeling large amounts of training data. Combined unsupervised/supervised training can simplify data collection and reduce expensive hand labeling.

4.1.1 Back-Propagation Classifier

Back-propagation classifiers form nonlinear discriminant functions using single- or multi-layer perceptrons with sigmoidal nonlinearities. They are trained with supervision, using gradient-descent training techniques, called back-propagation. Which minimize the squared error between the actual outputs of the network and the desired outputs. Patterns are applied to input nodes that have linear transfer functions. Other nodes typically have sigmoid nonlinearities. The desired output from output nodes is ''low'' (0 or < 0.1) unless that node corresponds to the current input class, in which case it is "high" (1 .*0* or > 0.9). Each output node computes a nonlinear discriminant function that distinguishes between one class and all other classes. Good introductions to back-propagation classifiers are available in many papers [21] and [22]. Early interest in back-propagation training was caused by the presupposition that it might be used in biological neural nets.

Figure 1 illustrates how the multi-layer perceptron can form three nonlinear input/output functions using back-propagation training. The multi-layer perceptron shown has n linear input node, p nodes with sigmoidal nonlinearities in the first hidden layer, and one linear output node.

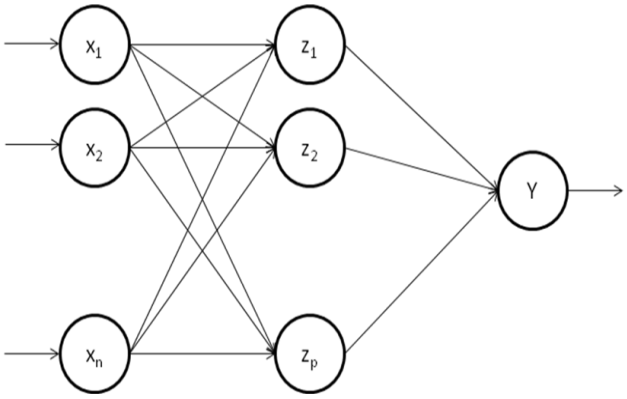


Figure 1: Multi Layer Feed Forward Artificial Neural Network

One major characteristic of back-propagation classifiers is long training times. Training times are typically longer when complex decision regions are required and when networks have more hidden layers. As with other classifiers, training time is reduced and performance improved if the size of the network is tailored to be large enough to solve a problem but not so large that too many parameters must be estimated with limited training data.

4.2 FLANN architecture

In this paper, a single layer model based on trigonometric expansion is presented. Let each element of the input pattern before expansion be represented as $x(i),\ 1<i<n$ where each element x(i) is functionally expanded as $z_j(i),\ 1<j<5$, where number of expanded points for each input element is 5 and n is total number of features in the dataset has been taken.

Expansion of each input pattern is done as follows.

$$\left.\begin{aligned} z_1(i) &= x_i, \\ z_2(i) &= \sin \pi(x_i), \\ z_3(i) &= \sin 2\pi(x_i), \\ z_4(i) &= \cos \pi(x_i), \\ z_5(i) &= \cos 2\pi(x_i) \end{aligned}\right\} \qquad (6)$$

where $z(i),\ 1<i<n$, n is the set of features in the dataset.

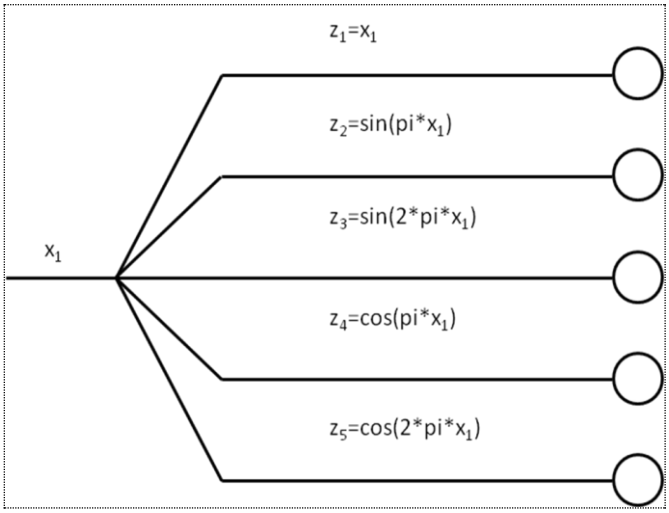


Figure 2: Functional Expansion of the First Element

.

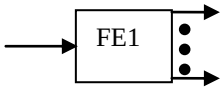


Figure 3: Block Form Representation of Figure 2

Figure 2 shows the functional expansion unit for one element of FLANN architecture and Figure 3 shows the block diagram of Figure 2. Figure 4 shows the architecture of the FLANN for classification.

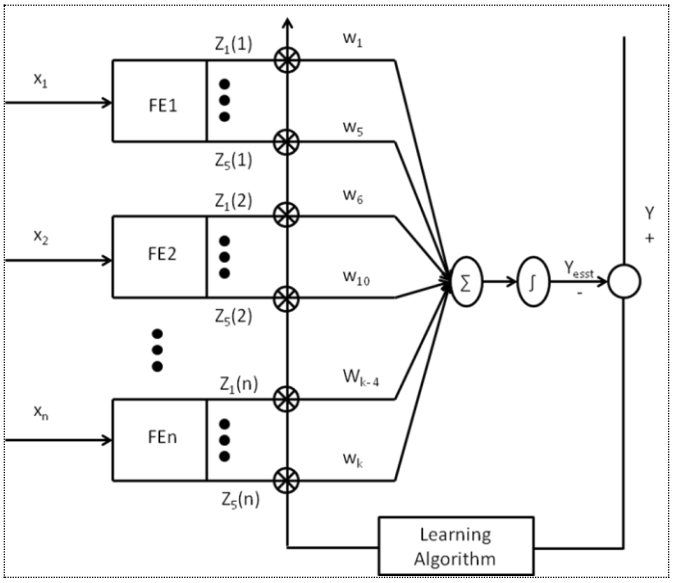


Figure 4: Architecture of FLANN

These nonlinear outputs are multiplied by a set of random initialized weights from the range [-0.5, 0.5] and then summed to produce the estimated output. This output is compared with the corresponding desired output and the resultant error for the given pattern is used to compute the change in weight in each signal path P [29].

5. SFLANN architecture

In SFLANN approach, PSO maintains a set of FLANNs as its particle. Each FLANN is evaluated for the training set. Figure 5 shows the architecture of the SFLANN. The FLANNs as particle, compete among themselves to obtain better weight set in the search space for designing the globally best FLANN model.

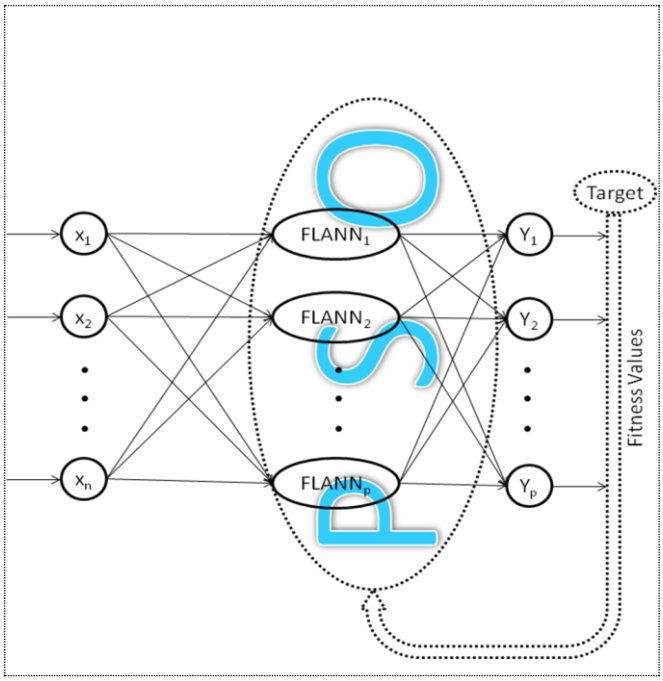


Figure 5: Architecture of SFLANN

In each iteration, each input pattern $< x_1, x_2, \ldots, x_n >$ from the training set is passed on to each FLANN maintained by the PSO. In the figure, p number of FLANNs are maintained by PSO, which produce $< y_1, y_2, \ldots, y_p >$ output for each input $< x_1, x_2, \ldots, x_n >$. After comparing these outputs with the respective target, error is evaluated. The squared error for each input for each FLANN is preserved. The sum up of errors for all the patterns in the training set for each FLANN is evaluated separately and treated as the negative fitness of the particle or FLANN, which is to be maximized by the PSO.

6. Simulation Results

The performance of different models is evaluated using the benchmark Life Science databases. All these databases are taken from the UCI machine learning repository [30]. Table 1 presents the summary of the main features of the datasets used for our experimental studies.

TABLE 1: Description of databases used

Dataset	*Patterns*	*Attributes*	*Positive Patterns*	*Negative Patterns*
Breast Cancer Wisconsin (Original)	699	9	458	241
Bupa Liver Disorders	345	6	145	200
Haberman's Survival	306	3	225	81
Horse Colic	300	27	181	119
Pima Indian Diabetes	768	8	268	500
Wilt	4339	5	4265	74

The training set is obtained from this database. Let total number of patterns is n, and then n numbers of random integers between 1 to n are generated. The random integers

generated more than once are not considered as part of this set. The unique integers form the index for selection of the training set from the original dataset. It is observed that about 65% of the total patterns are found in this process as the training set. Ten numbers of simulations are performed for each database. For each simulation a new training set is generated. From each simulation, the true positive, false negative, false positive and true negative values are obtained as in Figure 6.

		Predicted class: Positive	Predicted class: Negative	
Actual class	Positive	TP	FN	P
	Negative	FP	TN	N
		P'	N'	P+N

Figure 6: Confusion matrix

Figure 6 shows the confusion matrix, where, TP is true positive, FN is false negative, FP is false positive, TN is true negative, P is total actual positive, N is total actual negative, P' is total predicted positive, N' is total predicted negative and P+N is the total number of records. For each simulation, the classification accuracy is evaluated and the average values of the classification accuracy are presented in this section for analysis of results.

The protocol for parameters used for our simulation studies is given in Table 2.

TABLE 2: Parameters considered for simulation of SFLANN

Parameters	*Values*
Population Size	200
Maximum Iterations	500
Inertia Weight	0.729844
Cognitive Parameter	1.49445
Social Parameter	1.49445

The results of five different classifiers for 10 simulations of breast cancer database are presented in Table 3. The results shows that SFLANN model outperforms all other models in finding the maximum classification accuracy, minimum classification accuracy and the average classification accuracy of SFLANN is better than the maximum result of other models.

TABLE 3: Classification accuracy of 10 simulations for Breast Cancer database

Classifier models	*Average*	*Maximum*	*Minimum*	*Standard deviation*
SFLANN	**0.993**	**0.997**	**0.978**	0.00681
ANN	0.972	0.978	0.972	0.00457
Bayes	0.960	0.961	0.958	0.00117
KNN	0.974	0.977	0.968	0.00314
MLR	0.962	0.964	0.957	0.00245

Table 4 shows the results of five different classifiers for 10 simulations of bupa database. In this case KNN shows the best result, but SFLANN models yields the second best result.

TABLE 4: Classification accuracy of 10 simulations for Bupa database

Classifier models	*Average*	*Maximum*	*Minimum*	*Standard deviation*
SFLANN	0.735	0.762	0.709	0.01708
ANN	0.707	0.742	0.652	0.03059
Bayes	0.568	0.649	0.518	0.03857
KNN	**0.767**	**0.782**	**0.736**	0.01583
MLR	0.688	0.707	0.663	0.01312

The performance with Haberman database is presented in Table 5. In this case SFLANN model also yield better performance in comparison to other models.

TABLE 5: Classification accuracy of 10 simulations for Haberman database

Classifier models	*Average*	*Maximum*	*Minimum*	*Standard deviation*
SFLANN	**0.799**	**0.817**	**0.779**	0.01299
ANN	0.755	0.767	0.735	0.00942
Bayes	0.75	0.754	0.745	0.00317
KNN	0.769	0.79	0.751	0.01234
MLR	0.746	0.758	0.735	0.00728

The results of Horse colic database are presented in Table 6. Here ANN yields the best performance and SFLANN model gives second best performance.

TABLE 6: Classification accuracy of 10 simulations for Horse Colic database

Classifier models	*Average*	*Maximum*	*Minimum*	*Standard deviation*
SFLANN	0.836	0.86	0.81	0.01506
ANN	**0.9**	**0.926**	**0.873**	0.01633
Bayes	0.725	0.766	0.656	0.03493
KNN	0.833	0.863	0.81	0.01625
MLR	0.825	0.833	0.82	0.0045

The performance with Pima database is presented in Table 7. In this case SFLANN model also outperforms all other models.

TABLE 7: Classification accuracy of 10 simulations for Pima database

Classifier models	*Average*	*Maximum*	*Minimum*	*Standard deviation*
SFLANN	**0.818**	**0.852**	**0.790**	0.01919
ANN	0.651	0.651	0.651	0.0
Bayes	0.76	0.769	0.751	0.00592
KNN	0.799	0.811	0.776	0.01017
MLR	0.777	0.785	0.768	0.00525

The performance with Wilt database is presented in Table 8. Here KNN performs the best followed by ANN and Bayes. SFLANN and MLR yields the lowest performance, though the difference in performance is very small i.e. 0.80% .

TABLE 8: Classification accuracy of 10 simulations for Wilt database

Classifier models	*Average*	*Maximum*	*Minimum*	*Standard deviation*
SFLANN	0.983	0.983	0.983	0.00021
ANN	0.986	0.992	0.982	0.00412
Bayes	0.984	0.987	0.978	0.00273
KNN	**0.991**	**0.991**	**0.989**	0.00059
MLR	0.983	0.983	0.983	0.0

Out of the six databases taken, SFLANN model shows best performance in case of 3 databases, second best in case of 2 databases and competitive performance in one database.

7. Conclusions

A hybrid classifier model of PSO and FLANN called SFLANN is presented here. Life science databases are considered for evaluating the performance of SFLANN model. The performance of SFLANN is compared with ANN, Bayes, KNN and MLR classifiers. From the study it is revealed that the overall performance of SFLANN is better in comparison to other classifier models in majority of the problems.

References

[1] R. Agrawal, T. Imielinski, and A. Swami, “Database Mining: A Performance Perspective,” IEEE Transactions on Knowledge and Data Engineering, 5(6):914-925, December 1993.

[2] M. James, Classification Algorithms, Wiley, 1985.

[3] D. E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, Morgan Kaufmann, 1989.

[4] L. Breiman, J. H. Friedman, R. A. Olshen and C. J. Stone, Classification and Regression Trees, Wodsworth, Belmont, 1984.

[5] J. R. Quinlan, C4.5: Programs for Machine Learning, Morgan Kaufman, 1993.

[6] R. Lippmann, “An Introduction to Computing with Neural Networks,” IEEE ASSP Magazine, 4(22), April 1987.

[7] Y.-H. Pao, S. M. Phillips and D. J. Sobajic, “Neural-net computing and intelligent control systems,” *Int. J. Contr.*, vol. 56, no. 2, pp. 263–289, 1992.

[8] F. Heppner and U. Grenander, "A stochastic nonlinear model for coordinate bird flocks," In: Krasner S (eds.) The Ubiquity of Chaos. AAAS Publications, Washington, DC, 1990.

[9] C. W. Reynolds, "Flocks, herds, and schools: A distributed behavioral model,” Computer Graphics, vol. 21, no. 4, pp.25–34, 1987.

[10] R. C. Eberhart, P. Simpson and R. Dobbins, "Computational intelligence PC tools," Academic Press, 1996.

[11] E. O. Wilson, "Sociobiology: the new synthesis," Belknap Press, Cambridge, MA, 1975.

[12] J. Kennedy, and R. C. Eberhart, "Particle swarm optimization," Proceedings IEEE International Conference on Neural Networks IV, Piscataway, NJ, pp. 1942–1948, 1995.

[13] W. T. Reeves, "Particle systems – a technique for modeling a class of fuzzy objects," ACM Transactions on Graphics vol.2, no.2, pp.91–108, 1983.

[14] P. J. Angeline, "Evolutionary optimization versus particle swarm optimization: philosophy and performance differences," In: Porto VW, Saravanan N,Waagen D and Eiben AE (eds.). Evolutionary Programming VII, pp. 601–610, 1998.

[15] R. C. Eberhart, and Y. Shi, "Comparison between genetic algorithms and particle swarm optimization," In: Porto VW, Saravanan N, Waagen D and Eiben AE (eds.). Evolutionary Programming VII, pp. 611–616. Springer, 1998

[16] Y. Shi, and R. C. Eberhart, "Parameter selection in particle swarm optimization," In: Porto VW, Saravanan N, Waagen D and Eiben AE (eds) Evolutionary Programming VII, pp. 611–616. Springer, 1998

[17] Y. Shi, and R.C. Eberhart, "A modified particle swarm optimizer," Proceedings of the IEEE Conference on Evolutionary Computation. AK, Anchorage, 1998

[18] J. Kennedy, "The behavior of particles," In: Porto VW, Saravanan N, Waagen D and Eiben AE (eds.). Evolutionary Programming VII, pp. 581–590. Springer, 1998.

[19] Carlisle, and G. Dozier, "An off-the-shelf PSO," Proceedings of the Particle Swarm Optimization Workshop, pp. 1–6, 2001.

[20] U. Fayyad and R. Uthurusamy, "Data mining and knowledge discovery in databases," Communications of the ACM, vol. 39, pp. 24-27, 1996.

[21] W. H. Inmon, "The data warehouse and data mining," Communications of the ACM, vol. 39, pp. 49-50, 1996. October 1998.

[22] S. Mitra, S. K. Pal, P. Mitra, "Data Mining in Soft Computing Framework: A Survey", IEEE Transactions on Neural Networks, vol. 13, no. 1, January 2002.

[23] S. Bandyopadhyay, S.K. Pal, B. Aruna, "Multiobjective GAs, Quantitative, Indices, and Pattern Classification,” IEEE transactions on Systems, Man, and Cybernetics, Part-B, Vol.34, No.5, October 2004.

[24] R. P. Lippmann, "Pattern Classification Using Neural Networks", IEEE Communications Magazine, pp: 47-64, Nov 1989.

[25] R. O. Duda and P. E. Hart, “Pattern Classification and Scene Analysis”, NY: John Wiley and Sons, 1973.

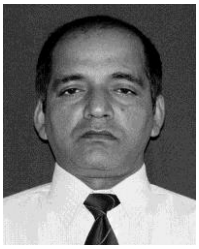
[26] K. Fukunaga, “Introduction to Statistical Panern Recognition”, NY: Academic Press, 1972.

[27] G. Nagy, "Candide's Practical Principles of Experimental Pattern Recognition," IEEE Trans. on

Panern Analysis and Machine Intel., vol. PAMI-5(2), pp. 199-200, 1983.
[28] J. R. Quinlan, "Improved Use of Continuous Attributes in C4.5," J. Artif. Intell. Res., vol. 4, pp. 77–90, 1996.
[29] B. B. Misra, S. Dehuri, "Functional Link Artificial Neural Network for Classification Task in Data Mining," J. Computer Sci., 3 (12): 948-955, 2007.
[30] C. L. Blake and C.J. Merz, "UCI Repository of machine learning databases," http://archive.ics.uci.edu/ml/.

Author Profile

Lokanath Satapathy received Master in Computer Application from Sambalpur University in 2010. During 2007-2009 he worked for the Koustuv institute of Self Domain, Bhubaneswar. Currently he is working at School of Computer Application, Kalinga Institute of Industrial Technology University, Bhubaneswar.



B. B. Misra has completed his B. Text. degree in 1984 from Kanpur University, M. Tech. (Computer Science) in 2002 from Utkal University, and Ph.D. (Engineering) in 2011 from Biju Pattanaik University of Technology. He has done his Post Doctoral Research during 2013-14, at AJOU University, South Korea under the Technology Research Program for Brain Science of Yonsei University. His areas of interests include data mining, soft computing, wireless sensor network, bioinformatics. He has published two books, three book chapters, and more than 65 papers in different journals and conferences of National and International repute. Presently he is continuing as Dean Research at Silicon Institute of Technology, Bhubaneswar.

B. N. Biswal received his B. Tech. degree in Electronics and Communication Engineering from Marathwada University in 1992, M.Tech. degree in Comunication and System Engineering from UCE Burla in 2001, Ph. D. degree in Computer Science from Utkal University in 2013. His areas of interest are Signal Processing, Data Mining, Computer Network, Architecture, VLSI, Database distributed systems etc. He has published one book and more than 10 research papers in journals and conferences of National and International repute. Currently he is working as the Director (A&A) for the Bhubaneswar Engineering College, India.