TRAINING FEED-FORWARD ARTIFICIAL NEURAL NETWORKS FOR PATTERN-CLASSIFICATION USING THE HARMONY SEARCH ALGORITHM

Ali Kattan
IT Department, College of Science
Ishik University
100 Meter Street, Erbil, Iraq
ali.kattan@ishikuniversity.net

Rosni Abdullah
School of Computer Sciences
Universiti Sains Malaysia
11800 Penang, Malaysia
rosni@cs.usm.my

ABSTRACT

The Harmony Search algorithm is relatively a young stochastic meta-heuristic that was inspired from the improvisation process of musicians. HS has been successfully applied as an optimization method in many scientific and engineering fields and was reported to be competitive alternative to many rivals. In this work a new framework is presented for adapting the HS algorithm as a method for the supervised training of feed-forward artificial neural networks with fixed architectures. Implementation considers a number of pattern classification benchmarking problems and comparisons are made against the traditional Back Propagation training method and an evolutionary based genetic algorithm training method. Results show that the proposed Harmony Search based method has attained results that are on par or better than those of Back Propagation and Genetic Algorithm. However BP seems to have better fine-tuning capabilities than the proposed HS-based method but might take longer overall training time.

Keywords: harmony search, feed-forward neural network, pattern classification, supervised training.

1. INTRODUCTION

The Harmony Search (HS) algorithm is a relatively young evolutionary stochastic global optimization (SGO) method [1]. This method draws its inspiration not from biological or physical processes but from the improvisation process of musicians. As an optimization method, HS was reported to be a competitive alternative to other SGO methods [2] and has been applied successfully in many applications in engineering and industry [3-7]. A significant amount of research has already been carried out on the application of HS for solving various optimization problems [2, 4, 5, 8, 9]. The search-mechanism of HS has been explained analytically within a statistical-mathematical framework [10] and HS as an SGO method is being compared against other evolutionary based methods such as genetic algorithm (GA) [2, 4]. Feed-forward artificial neural networks (FFANN) are considered to be powerful tools in the area of pattern classification [11]. A universal FFANN approximators for arbitrary finite-input environment measures can be built using only a single hidden layer [12]. Using a dataset composed of patterns along with the target value of each, the supervised training process of an artificial neural network (ANN) is concerned with adjusting the individual weights between each of the individual ANN neurons until we can achieve the the desired output. Supervised ANN training would usually require numerous cycles where training data is used by an algorithm to adjust the weights accordingly [13]. Evolutionary ANN training models such as GA is the result of combining an evolutionary optimization algorithm with the ANN learning process [14]. Search features of these learning models contrast with those of the standard gradient-descent Back Propagation (BP) in that it is not trajectory-driven, but population driven. They overcome many of the inefficiencies of trajectory-driven methods such as local minima entrapment by promoting exploration of the search space [15].

In this work a framework is presented to adapt and model the HS algorithm as new evolutionary-based training method for FFANNs with fixed architectures. Experiments are carried out on a set of pattern classification benchmarking problems and comparisons are made against BP and GA based training methods. It is worth to mention that there are many HS variants that have been already introduced in the literature [16-18]. These variants basically propose different enhancements to that of the original HS algorithm proposed by Lee and Geem [6], which is referred to as “classical” [10]. This work considers adapting the classical HS only.

This paper is organized as follows: section 2 presents some related works. Section 3 presents the basic HS algorithm and explains its main concepts. Section 4 introduces the proposed

method and section 5 presents the experimental results. The conclusions are given in section 6.

2. RELATED WORKS

The BP learning method became a popular method to train FFANN [19, 20]. The algorithm is a trajectory-driven technique that is analogous to an error-minimizing process. In spite of the fact that trajectory-driven training techniques date back to more than three decades, many recent works still consider such training methods [21-26]. BP learning requires the neuron transfer function to be differentiable and suffers from the possibility of falling into local minima. The method is also known to be sensitive to the initial weight settings where many weight initialization techniques have been proposed to lessen such a possibility [27-29].

The training of FFANNs using evolutionary-based training model is the result of combining an evolutionary optimization algorithm such as GA [30], ant colony optimization [31], particle swarm optimization [32] and improved bacterial chemo-taxis optimization [33], with the ANN learning process. Search features of these learning models contrast with those of the standard gradient-descent BP in that it is not trajectory-driven, but population driven. Using a suitable network representation, the process of supervised training using an evolutionary method involves performing several iterations in order to minimize or maximize a certain fitness function [34-36]. Such optimization process would usually stochastically generate vectors representing the network's weight values including biases, calculate the fitness for the generated vectors and tries to keep those vectors that give better fitness values in that model's population. It is also possible to include the network structure in such representation where the structure can also evolve [32]. The cycle is repeated to generate new offspring and eventually after several iterations the training process is halted based on some criteria [37]. The optimal or near optimal solution is given by the best member of population, characterized by having the best fitness value.

Several types of fitness functions have been used in the evolutionary ANN supervised training models. The common factor between all of these fitness functions is the use of ANN output error where the goal is usually to minimize such error. This is the difference in value between the actual and expected output and it can be represented in different forms. The most common fitness functions are those that are based on the sum of square errors (SSE) or the mean square error (MSE) formula [30, 34, 36, 38-41]. SSE has also been used as a criterion function in trajectory-driven supervised training methods such as BP [13, 27].

Dorsey et al [34] have designed a Genetic Adaptive Neural Network Training (GANNT) algorithm that uses SSE as the main fitness function . GANNT is still being referenced and used in some relatively recent works [42, 43]. Instead of using a binary string representation for the population members, the FFANN inter-node weights are combined to form a real-value vector, or a string, representing the chromosome such that each weight value of this string is an atomic unit. It has been shown that the binary representation is neither necessary nor beneficial and it limits the effectiveness of the GA [44]. Each population string in this case would represent a complete set of FFANN weights. Experiments were conducted on several problems implemented using a fixed 3-layer FFANN structure. Alba et al [35] has conducted a comparative study on some GA hybrid FFANN training methods against two gradient descent methods. He used the ANN Squared Error Percentage (SEP) and the Classification Error Percentage (CEP) as two fitness functions indicating that both should be reported in the field of ANN research. Experiments were also conducted on a 3-layer fixed FFANN pattern classifier. Kim et al [36] presented a modified GA for fast training FFANN. He represented the FFANN as weighted digraph, with no closed paths, described by an upper or lower adjacency matrix with real valued elements. Nodes are supposed to be in a fixed order according to layers. The population would contain a number of such adjacency matrices and the GA crossover is to be carried out row wise or column wise on these matrices.

3. THE HARMONY SEARCH ALGORITHM

The HS algorithm is a meta-heuristic SGO method similar in concept to other SGO methods in terms of combining the rules of randomness to imitate the process that inspired it. The method can handle discrete and continuous variables with similar ease [6]. HS concept is based on the improvisation process of musicians in a band. Improvisation

occurs when each musician tests and plays a note on his instruments such that the resultant tones are considered by an aesthetic quality measure as in harmony with the rest of the band. Each instrument would have a permissible range of notes that can be played representing the pitch value range of that musical instrument.

To improvise a new harmony, each musician would either play a totally new random note from the permissible range of notes, play an existing note from memory, or play a note from memory that is slightly modified. Only good improvised harmonies are kept and remembered by musicians till better ones are found and replace the worst ones. Each note played by a musician represents one component of the solution vector of all musician notes and as shown in Figure (1). The perfect solution vector is found when each component value is optimal based on some objective function evaluated for this solution vector [45].

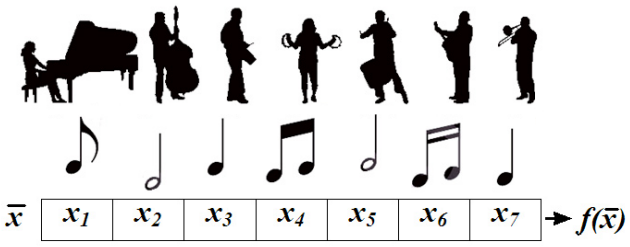

Figure 1. Music improvisation process for a harmony in a band of seven

The Harmony Memory (HM) is a matrix of the best solution vectors attained so far. The harmony memory size (HMS) is set prior to running the algorithm. The number of components in each harmony vector N is analogous to the tone's pitch, i.e. note, values played by N musical instruments. N represents the total number of decision variables. If continuous decision variables are considered then each pitch value is drawn from a pre-specified permissible range of values. The ranges' lower and upper limits are specified by two vectors x^L and x^U both having the same length N. Each harmony vector is also associated with a harmony quality value (i.e. fitness) based on an objective function f(x). The modeling of HM is shown in Figure (2).

In order to improvise a new vector (harmony), each decision variable (instrument) is considered separately. HS uses two probabilistic parameters to reflect playing choices. These are the Harmony Memory Considering Rate (HMCR) and the Pitch Adjustment Rate (PAR). The former determines the probability of selecting a value (playing a pitch) from memory or selecting a totally new random one drawn from the permissible range for that decision variable. The latter, PAR, determines the probability of whether the value that is selected from memory is to be adjusted or not. The adjustment value for each decision variable is drawn from the respective component of a Bandwidth vector **B** having the size N. The adjustment process should guarantee that the resultant value is within the permissible range specified by x^L and x^U. The pseudo code for the classical HS algorithm is given in Figure (3). The HS algorithm would require the initialization of a number of parameters. These are HMS, HMCR, PAR as well as the vectors **B** and $[\mathbf{x}^L, \mathbf{x}^U]$. In addition, a maximum number of improvisations (MAXIMP) should be set for which the algorithm terminates.

4. THE PROPOSED METHOD

HS is being compared to other evolutionary SGO methods in particular GA were the FFANN training techniques of the latter have been discussed thoroughly in the literature and comparisons are made against trajectory-driven methods such as BP [38-40, 42, 44, 46]. In GA-based training methods, the training process translates to using suitable FFANN weights representation, fitness function and termination condition(s).

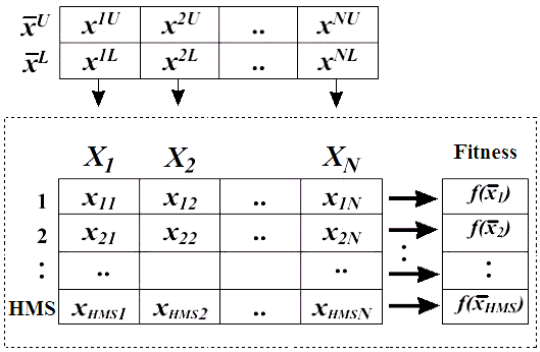

Figure 2. The modeling of HM with N decision variables

Initial parameter settings are also required. Inspired by these, HS could be used as an FFANN training method by adapting it to handle these three issues such that they suit the functionality of the HS algorithm and satisfy the requirements

mandated by the FFANN supervised training process. Each of these issues is addressed below and an adapted HS-based FFANN training algorithm is proposed.

FFANN Data Representation

Considering the concept behind the GA-based training methods, harmony vectors can be thought of as being the network's weights where each vector represents a complete set of FFANN weights. As presented earlier in section 2, there are two common candidate real-valued FFANN weight representations that could be utilized; the vector-based representation [34] and the adjacency matrix-based representation [36]. The former vector-based representation lends itself more suitably for the HS algorithm. However, the algorithm must be adapted to deal with common ranges specified by vectors [$\mathbf{x}^{\mathbf{L}}$, $\mathbf{x}^{\mathbf{U}}$] and **B** since FFANN weights have common value ranges and are not discrete decision variables of an optimization problem. To adapt for such situation, the HM representation in this case can be thought of as having different musicians all of whom are using the same musical instrument. Using the same musical instrument will imply that the instruments will have a common pitch range. Thus the component values of both [$\mathbf{x}^{\mathbf{L}}$, $\mathbf{x}^{\mathbf{U}}$] and **B** are the same for all decision variables in this case and there will be no need to use these two vectors. These can be simply replaced by the scalar range [x^L, x^U] and the value B. The adapted HM representation is shown in Figure (4) where the vector W represents the improvised FFANN weight values.

Each harmony vector in HM is represented using the vector representation from the GANNT algorithm [34] and as illustrated in Figure (5). The vector represents a complete set of FFANN weights including biases. Neurons respective weights are listed in sequence assuming a fixed FFANN structure.

Step 1: Initialize the algorithm parameters (HMS, HMCR, PAR, B, MAXIMP) and specify the optimization problem as *Minimize f(x)* subject to $x_i \in X_i, i = 1,2,...,N$

Step 2: Initialize the harmony memory HM with random values drawn from vectors [$\mathbf{x}^{\mathbf{L}}$,$\mathbf{x}^{\mathbf{U}}$].

Step 3: Improvise new harmony vector $\bar{x}_i$:

For each vector component

$$x_i' \leftarrow \begin{cases} x_i' \in \{x_{i1}, x_{i2}, .., x_{iHMS}\} & \text{with probability HMCR} \\ x_i' \in X_i & \text{with probability (1-HMCR)} \end{cases}$$

If probability HMCR, do Pitch adjusting

$$x_i' \leftarrow \begin{cases} x_i' \pm rand(0,1) \cdot B_i & \text{with probability PAR} \\ x_i' & \text{with probability (1-PAR)} \end{cases}$$

$$x_i' \leftarrow \min(\max(x_i', x_i^L), x_i^U) \quad \text{//bounds check}$$

Step 4: If the newly improvised harmony $\bar{x}_i$ is better than the worst harmony in HM then replace it.

Step 5: Terminate if the current improvisation >MAXIMP. Otherwise, repeat step 3 & 4.

Step 6: Best solution is given by the best harmony in HM.

Figure 3. The classical HS algorithm

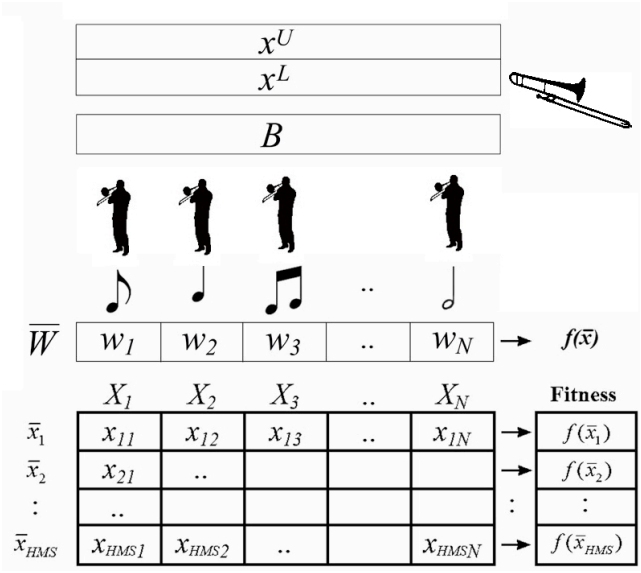

Figure 4. Adapted HM representation for FFANN training

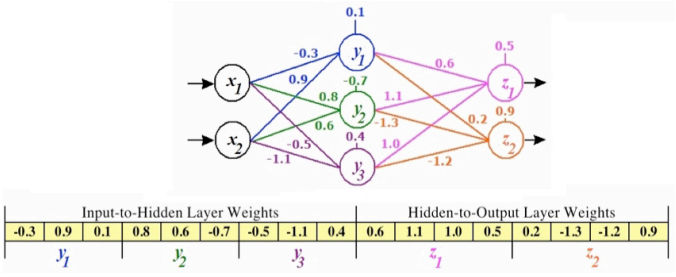

Figure 5. Weight vector representation of a sample FFANN

4.2 Fitness Function

With SSE being one of the most commonly used fitness functions for evolutionary-based supervised training, it is selected as main quality measure for the proposed HS-based training algorithm. Since this work considers pattern-classification problems, CEP is selected as a second quality measure to complement the raw error values given by SSE and report in a high-level the quality of the trained network. SSE, as the main fitness, is the measure on which the HM vectors are to be sorted on from best to worst with the best being the harmony with the least SSE value. Thus for an improvised harmony, i.e. FFANN weight vector, to be accepted, its SSE value must be smaller than the worst one in the current HM. SSE could be used as the sole fitness function in the algorithm. However, considering the possibility that lower SSE values do not necessarily give a better classification ability, CEP is also used to lessen such possibility and would be computed for each harmony vector. The justification can be explained by considering the winner-take-all approach used for the pattern-classification problems considered. FFANNs used for pattern-classification have more than one output unit in its output layer to designate "classes" or "groups" belonging to a certain type [12, 47]. The unit that produces the highest output among other units would indicate the winning class, a technique that is known the "winner-take-all" [27, 48]. Lower CEP values are not necessarily associated with lower SSE values. This stems from the fact that even if the SSE value is small, it is the winner class, i.e. the output neuron with the highest value, which determines the result of the classification process.

In order to accept the newly improvised harmony, a condition based on its SSE and CEP value must be devised. A condition whereby the newly improvised FFANN weight vector is to be less in terms of both SSE and CEP from that of the worst in HM is too stringent and might take long time to converge or obtain results far from optimal. Such conclusion was based on some early empirical tests in the early stages of this work. A less stringent quality test would be for the newly improvised FFANN weight vector to have SSE value less than that of the worst in HM and a CEP value that is less than or equal to the average CEP value of HM. Both of these quality measures are to ensure that only better harmonies are to replace worst ones.

In order to compute SSE and CEP, forward-pass calculations must be performed on the given FFANN structure. This is a repetitive process that involves loading the whole training dataset. This would require a process by which the network weights, represented by the harmony vector, are to be loaded into the FFANN to perform such computation. The FFANN architecture must be therefore flexible to allow loading different weight vectors during the HS algorithm initialization and improvisation processes. SSE, given in (1), will be associated with each HM vector to represent its harmony quality. The forward-pass calculations for each neuron involve finding the sum of input signals and then applying the neuron transfer function. These are given in (2) and (3) using the bipolar sigmoid as the neuron transfer function [27].

$$SSE = \sum_{p=1}^{P}\sum_{i=1}^{S}(t_i^p - z_i^p)^2 \quad (1)$$

where
P : total number of training patterns
S : total number of output units (classes)
t : target output
z : actual output

$$y = \sum_{i=1}^{n+1} w_i x_i \quad (2)$$

$$z = F(y) = \frac{2}{1+e^{-y}} - 1 \quad (3)$$

where
x_i : input value from unit ***i*** of previous layer (output of that unit)
w_i : the weight between this neuron and unit ***i*** of previous layer (w_{n+1} represents bias)
y : sum of the neuron's input signals
n+1: total number of input connections including bias
$F(y)$: neuron transfer function (bipolar sigmoid)
z : neuron output

CEP is the percentage of incorrectly classified patterns for the whole training dataset and as given in (4). CEP would be used to complement SSE raw error values since CEP reports in a high-level manner the quality of the trained ANN [35].

$$CEP = \frac{E_P}{P} \cdot 100\% \quad (4)$$

where
E_P: total number of incorrectly recognized training patterns
P : total number of training patterns

In the proposed implementation SSE is used as the main fitness function assisted by CEP as a second fitness measure. If the newly improvised harmony vector has a lower SSE value than the worst harmony in HM and its calculated CEP value is lower or equal to the average CEP of HM, then the newly improvised harmony vector is accepted and inserted in HM discarding the current worst vector.

4.3 Termination Condition

Training termination in HS is determined solely by the value of MAXIMP. The choice of this value is a subjective issue that is based on experience and has nothing to do with the quality of the best-attained solution.

4.4 HS Initial Parameters Settings

The classical HS involves the initialization of certain number of parameter values before running the algorithm. These are HMS, HMCR, PAR, [x^L, x^U], B and MAXIMP. The initial value setting for each is discussed below from an FFANN training perspective.
Based on the fact that HM resembles the short-term memory of musicians, the use of small values for HMS is recommended where previous findings showed an independence to the value of HMS and that no single choice is superior to others [17]. Considering the FFANN data representation introduced earlier, a higher value of HMS would increase the overall computation time since the calculations of the fitness function would be required for each HM vector during the initialization process. The overall computation time for each vector would be proportional to the total number of FFANN weights and the total number of training patterns. The value HMS=10 was encountered in many parameter estimation problems [9, 49]. A smaller value of HMS=5 was used for some integer programming problems [17]. Higher values are also used [6, 50]. The HMS value selection should consider the total number of FFANN weights as well as the size of the training dataset. Considering evolutionary-based FFANN training, it was indicated that a population size of 10 is large enough for most FFANN problems to get good results [51].
For HMCR, several HS optimization problems used the value of 0.9 or higher [6, 50, 52]. Based on the work in [17], it is recommended to use large values for HMCR such that HMCR ≥0.9 if the problem has high dimensionality, which translates to the total number of FFANN weights in this case. The lengths of the decision vectors considered in this work are larger than those used for the aforementioned HS optimization applications.
PAR and B are the key parameters in the improvisation process. PAR values ranging between 0.3 and 0.45 were encountered in many

applications [6, 45, 52]. It is recommended to use relatively small values for PAR such that PAR≤0.5. [17]. The bandwidth vector **B** draws values from permissible ranges represented by the two vectors [$\mathbf{x}^{\mathbf{L}}$, $\mathbf{x}^{\mathbf{U}}$]. Since FFANN weight values have a common range as discussed earlier, fixed ranges are used for all decision variables instead for the vectors $\mathbf{x}^{\mathbf{L}}$, $\mathbf{x}^{\mathbf{U}}$ and **B**. Thus we would have the scalar range [x^L, x^U] and the scalar value B. The latter specifies the range of permissible weight adjustments given by the range [-B,B]. These parameters would depend on the permissible weight value range used by the FFANN as well as the application.

Training termination in HS is determined solely by the value of MAXIMP. The choice of this value is a subjective issue and values like 5000, 10000 or higher were commonly used in many applications [17, 50, 52]. Higher values would usually generate more accepted improvisations. MAXIMP values are chosen based on experience and the nature of the application.

4.5 The Adapted HS-based Training Algorithm

The adapted classical HS training algorithm is given as pseudo code in Algorithm (1) and the standard HS harmony vector improvisation process is given in Algorithm 2. In Algorithm 1, initialization is performed in Line 1 and 2. Line 3 would involve iterating through the whole training set for each initial vector in order to compute the harmony vector's respective SSE and CEP. This is written as one line of code here for simplicity. Once the CEP values for all harmony vectors are obtained, the HM average CEP is computed in line 4. Line 6 through 21 is the actual HS stochastic iterative process to improvise weight vectors. The process involves improvising a new harmony vector, loading it into the FFANN and performing the feed-forward computations to find the respective SSE and CEP using the training dataset. The improvisation process, given in Algorithm (2), is made via a call in Line 7. Line 17 would test the quality of the new improvised vector as discussed earlier. If the new harmony is accepted it will replace the worst vector in the current HM and a new average CEP of the HM is computed. The whole process is then repeated until termination occurs once the maximum number of iterations is reached as specified by MAXIMP value.

5. EXPERIMENTS

In order to demonstrate the performance of the proposed method, four different pattern-classification benchmarking problems were obtained from the UCI Machine Learning Repository[1] [53] for the experimental testing and evaluation as given in Table (1). These problems are taken from different fields including biology, engineering, forensic science and medical research. One of the main reasons behind choosing these problems is that they had no missing values in their given datasets. In addition these problems have been commonly used in the literature addressing different aspects such as classification techniques, recognition accuracy, overall training time, best architecture, etc. The last two datasets, namely Glass and Thyroid, are also characterized by having smaller class percentages compared to the rest of the class for that problem. For instance, Thyroid classes "hyber-function" and "subfunction" contain smaller number of patterns in comparison with the "normal" class. This is also true for the second, third and fourth classes of the Glass problem. The aim in this case is to test the fine-tuning capabilities of the proposed training method.

A 3-layer FFANN was designed for each problem to work as a pattern classifier. The output layer has a number of output neurons that is equal to the number of classes used in each problem. Each pattern in the dataset is associated with a target integer value representing the target class. The winner-take-all fashion [13, 35] was used to indicate the FFANN output winning class. All neurons use the bipolar sigmoid transfer function and the datasets values were normalized to be in the range of [-1, 1]. A common practice is to split the dataset to use 80% of the patterns as training set and the rest for post training out-of-sample testing set [31, 54]. The training and testing sets were made to include equal percentages of each class. The selected classification problems given in Table (1) are listed in an ascending order based on the number of weights required by each problem's FFANN structure.

The initial parameter settings for the BP, GA and the proposed HS-based training methods are summarized in Table (2). For BP a low learning rate value was used since such value would

[1] For datasets downloads and their full citations see http://archive.ics.uci.edu/ml

generally achieve a better trained network [55]. The Nguyen-Widrow weight-initialization technique was adopted to lessen the possibility of converging to a local minimum [27]. For GA, an implementation of GANNT [34, 42, 43] was utilized with the parameters values suitable for the problems considered in this work. HS initial parameter values were based on the discussions of the previous section. Both GA and HS would have the same population size in this case. The $[x^L, x^U]$ range is selected to give a bigger search space for HS-based method to explore which is also taken as the same initialization range for GA. The B value was determined based on several independent tests considering the selected $[x^L, x^U]$ range values. Since termination in HS depends entirely on the subjective selection of MAXIMP, two values were investigated, a low value of 5000 and a higher value of 20000. The program creates a log file during run-time to record the algorithm relevant parameter values upon each accepted improvisation. Java 6 was used for implementation and all experiments were carried out on the same computer. Ten independent training sessions for each of the selected problems were run for each training method. The best out of these ten are those solutions that give the highest overall recognition ratio in the least amount of total training time and these are listed here in Table (3) and Table (4). Some of the table fields are "not applicable" to BP and GA where those are marked as (N.A.). The "Total Accepted" column for BP and GA represents the actual number of iterations performed by the method. CEP is calculated in BP but it is not used in the output error calculations for this method and serves only as a comparison measure.

In spite the fact that the attained SSE and CEP values of BP for each of the four problems are lower than those obtained by the proposed method, the proposed method performed better in terms of the percentages of the overall correctly recognized classes and the overall training time in three out of four problems. The lower SSE values achieved by some problems have resulted in losing some of the network's generalization ability due to overtraining. This is caused by using too many training cycles as specified by MAXIMP causing the network to adjust it weighs more specifically to the training set and loses its generality [56].

In terms of the overall recognition percentage, results are close or better than those achieved using BP and GA. In general, a higher MAXIMP value would yield more accepted improvisations and as indicated by the "Total Accepted" column in the results' tables. Checking the results tables in terms of the selected MAXIMP value and the overall recognition percentage, it is obvious that choosing a higher value for some problems would result in attaining better results such as the case with the Ionosphere problems while it is the opposite for the other problems where overtraining is exhibited.

The overall recognition percentages obtained by the proposed HS-based training method is either better or close to those obtained by BP and GA. For those problems where HS didn't perform best the overall training time for HS is less in comparison with the others. BP has scored higher in the Thyroid problem although it took much longer overall training time. GA on the other hand has achieved a close overall recognition percentage with the least time among all but at the expense of having poor fine-tuning capabilities in terms of the classes recognition ratios. BP seems to have better target classes' recognition percentages in both the Glass and the Thyroid problems.

Some of the target classes of these two problems have relatively low percentages in their training datasets compared to others as given earlier in Table (1). The HS-based method has scored zero percent recognition for some of the target classes in these two problems, while BP maintains much higher values. BP is a local search method that can better fine-tune the final result even if the number of training patterns is relatively smaller. Having a larger total number of training patterns, as the case with Thyroid, would give the additional advantage of refining the solution. The proposed HS-based training method seem to lack such fine-tuning capability considering such problems but can attain a close overall recognition percentage in less overall training time.

6. CONCLUSIONS

The supervised training of FFANN using the adapted classical HS algorithm gave better or close overall recognition percentages than the same FFANN trained using the BP or GA based training methods. However, BP seems to have better fine-tuning capabilities in FFANN classification problems with higher number of classes. Such

fine-tuning capabilities are more apparent in problems having large datasets giving BP an extra advantage to further fine-tune the solution. The maximum number of improvisations determines the FFANN training convergence state for the proposed HS-based method where this is the sole termination condition in HS. The choice of this value is a subjective issue and large values could result in overtrained networks while smaller values might miss a better solution.

Future work should consider devising a better way of determining the convergence state based on the quality of the attained best solution. Hybridizing HS with another local search method could be utilized to detect such convergence state and achieve better fine-tuning capabilities. Such technique has been already used in many other evolutionary FFANN training rivals and is inevitable to use in order to enhance the proposed method. If larger FFANN with larger datasets are to be considered then the intrinsic parallel nature of FFANN calculations would invite the use of a parallel implementations to speed up the fitness function calculations resulting in a reduction in the overall training time required by HS.

ACKNOWLEDGMENT

This research is supported by UNIVERSITI SAINS MALAYSIA and has been funded by the Research University Cluster (RUC) grant titled by "Reconstruction of the Neural Microcircuitry or Reward-Controlled Learning in the Rat Hippocampus" (1001/PSKBP/8630022).

```
1  Initialize the algorithm parameters
2  Initialize HM with random weight values drawn
   from [x^L,x^U]
3  Do feed-forward computation for each vector
   to find SSE and CEP
4  Find HM average CEP
5  Begin
6    For imp=1 to MAXIMP do
7      Call: Improvise new harmony vector x
8      Load x weights into the FFANN
9      SSE=0, Class Error CE = 0
10     For each training pattern do
11       Apply training pattern to FFANN input
12       Compute the feed-forward phase,
         find the Error E
13       SSE= SSE + E^2
14       If the class is not recognized
         then CE = CE + 1
15     EndFor
16     CEP= CE/Total Patterns * 100
17     If x SSE < worst vector SSE AND
         x CEP <= HM average CEP then
18       Replace worst harmony vector with x
19       Recalculate HM average CEP
20     EndIf
21   EndFor
22   Best solution is given by the best
       harmony in HM
23 End
```

Algorithm 1. pseudo code for the HS-based FFANN supervised raining

```
1    Create new harmony vector x' of size N
2    For i=0 to N do
3      RND = Random(0,1)
4      If (RND<=HMCR)
       //harmony memory consideration
5        RND = Random(0,HMS)
6        x'(i) = HM(RND,i)
         //harmony memory access
7        RND = Random(0,1)
8        If (RND<=PAR) //Pitch Adjusting
9          x'(i) = x'(i) + Random(-B,B)
10         x'(i) = min(max(x'(i),x^L),x^U)
11       EndIf
12     Else //random harmony
13       x'(i) = Random(x^U,x^L)
14     EndIf
15   EndFor
16   Return x'
```

Algorithm 2. pseudo code for improvising new harmony vector

Table 1. Dataset Features

Dataset Name	FFANN Structure	Total Weights	Patterns		Total Class Percentages (Class index: percentage)
			Training	Testing	
IRIS ***Iris Dataset:*** Predict class of iris plant based on certain plant measurements.	4-5-3	43	120	30	0: 33.33% setosa 1: 33.33% versicolour 2: 33.33% virginica
IONOSPHERE ***Ionosphere Dataset:*** Classification of radar returns from the ionosphere corresponding to the complex values returned by the function resulting from the complex electromagnetic signal. The original dataset had 34 input features; however the third feature is a constant value of 9.0 for all instances and therefore removed.	33-4-2	146	280	71	0: 64:10 good 1: 35.90 bad
GLASS ***Glass Identification Dataset:*** From USA Forensic Science Service where 6 types of glass defined in terms of their oxide content. The original dataset had 7 classes but the dataset did not include any instance of this 7th class and thus only 6.	9-12-6	198	171	43	0: 32.71% f. building 1: 35.51% non. f. building 2: 7.94% vehicle 3: 6.07% containers 4: 4.21% tableware 5: 13.55% headlamp
THYROID ***Thyroid Disease Dataset:*** Detect thyroid function based on certain biological features.	21-15-3	378	5,760	1,440	0: 92.57% normal 1: 5.14% hyper-function 2: 2.29% subfunciton

Table 2. List of parameters used by training algorithms

M	Parameter	Values
BP	Learning Rate	0.008
	Momentum	0.7
	Initial Weights	[-0.5, 0.5]
	Initialization Method	Nguyen-Widrow
	Stopping Criterion	SSE difference<= 1.0E-4
GA	Population Size	10
	Crossover	At k=rand(0,N), no crossover for k=0
	Mutation Probability	0.01
	Value Range	[-250, 250]
	[min,max]	50% domination of a fitness value
	Stopping Criterion	
HS	HMS	10
	HMCR	0.97
	PAR	0.3
	B	5.0
	$[x^L, x^U]$	[-250, 250]
	MAXIMP	5000, 20000

Table 3. Results for Iris and Ionosphere datasets

Set	M	Training							Testing	
		MAXIMP	SSE	CEP	Total Accepted	Last Accepted Iteration #	Last Accepted Time	Overall Time	Overall Recog. %	Class Recog. %
IRIS	HS	5000	21.6	0.83%	127	4564	0:02:32	0:02:46	**96.67%**	100.00% 100.00% 90.00%
		20000	18	1.67%	162	2390	0:01:18	0:10:51	96.67%	100.00% 100.00% 90.00%
	BP	N.A.	7.85	0.83%	1254	N.A.	N.A.	0:07:29	96.67%	100.00% 100.00% 90.00%
	GA	N.A.	96	10%	66	N.A.	N.A.	0:00:34	90.00%	100.00% 90.00% 80.00%
IONOSPHERE	HS	5000	128	6.07%	151	4773	0:03:49	0:04:00	91.55%	100.00% 76.00%
		20000	106.4	5.00%	170	19463	0:20:46	0:21:20	94.37%	97.83% 88.00%
	BP	N.A.	8.52	0.56%	1628	N.A.	N.A.	0:24:43	**95.77%**	100.00% 88.00%
	GA	N.A.	152	6.79%	2244	N.A.	N.A.	0:35:57	94.37%	100.00% 84.00%

Table 4. Results for Glass and Thyroid datasets

Set	M	Training							Testing	
		MAXIMP	SSE	CEP	Total Accepted	Last Accepted Iteration #	Last Accepted Time	Overall Time	Overall Recog. %	Class Recog. %
GLASS	HS	5000	489.06	38.6%	108	4984	0:03:52	0:03:53	**76.74%**	92.86% 46.67% 0.00% 33.33% 0.00% 100.00%
		20000	355.86	29.82%	177	19798	0:20:18	0:20:31	72.09%	85.71% 80.00% 0.00% 0.00% 50.00% 100.00%
	BP	N.A.	218.06	18.71%	662	N.A.	N.A.	0:06:08	67.44%	35.71% 100.00% 33.33% 33.33% 50.00% 100.00%
	GA	N.A.	544	42.11%	6123	N.A.	N.A.	1:17:00	67.44%	85.71% 73.33% 0.00% 0.00% 0.00% 100.00%
THYROID	HS	5000	3211.2	7.24%	67	4829	2:37:43	2:43:18	92.78%	9.09% 0.00% 100.00%
		20000	3146.4	6.94%	94	19464	10:47:19	11:05:04	92.71%	9.09% 0.00% 99.92%
	BP	N.A.	450.23	1.33%	4201	N.A.	N.A.	22:11:47	**97.22%**	78.79% 68.92% 99.25%
	GA	N.A.	3416	7.42%	167	N.A.	N.A.	1:58:49	92.57%	0.00% 0.00% 100.00%

REFERENCES

[1] Z. W. Geem, J. H. Kim, and G. V. Loganathan, "A New Heuristic Optimization Algorithm: Harmony Search," *Simulation,* vol. 72, pp. 60-68, 2001.

[2] P. Tangpattanakul and P. Artrit, "Minimum-time trajectory of robot manipulator using Harmony Search algorithm," presented at the 6th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON 2009) Pattaya, Thailand, 2009.

[3] Z. W. Geem, *Music-Inspired Harmony Search Algorithm: Theory and Applications* vol. 191: Springer, 2009.

[4] J.-H. Lee and Y.-S. Yoon, "Modified Harmony Search Algorithm and Neural Networks for Concrete Mix Proportion Design," *Journal of Computing in Civil Engineering,* vol. 23, pp. 57-61, 2009.

[5] Z. W. Geem, "Harmony Search Applications in Industry," in *Soft Computing Applications in Industry*. vol. 226/2008, ed: Springer Berlin / Heidelberg, 2008, pp. 117-134.

[6] K. S. Lee and Z. W. Geem, "A New Meta-heuristic Algorithm for Continuous Engineering Optimization: Harmony Search Theory and Practice," *Computer Methods in Applied Mechanics and Engineering,* vol. 194, pp. 3902-3933, 2005.

[7] R. Forsati, A. T. Haghighat, and M. Mahdavi, "Harmony search based algorithms for bandwidth-delay-constrained least-cost multicast routing," *Computer Communications,* vol. 31, pp. 2505-2519, 2008.

[8] W. S. Jang, H. I. Kang, and B. H. Lee, "Hybrid Simplex-Harmony search method for optimization problems," presented at the IEEE Congress on Evolutionary Computation (CEC 2008) Trondheim, Norway, 2008.

[9] H. Ceylan, H. Ceylan, S. Haldenbilen, and O. Baskan, "Transport energy modeling with meta-heuristic harmony search algorithm, an

application to Turkey," *Energy Policy,* vol. 36, pp. 2527-2535, 2008.

[10] A. Mukhopadhyay, A. Roy, S. Das, S. Das, and A. Abraham, "Population-variance and explorative power of Harmony Search: An analysis," presented at the Third International Conference on Digital Information Management (ICDIM 2008) London, UK, 2008.

[11] U. Seiffert, "Training of Large-Scale Feed-Forward Neural Networks," presented at the International Joint Conference on Neural Networks, Vancouver, BC, Canada, 2006.

[12] X. Jiang and A. H. K. S. Wah, "Constructing and training feed-forwardneural networks for pattern classifcation," *Pattern Recognition,* vol. 36, pp. 853-867, 2003.

[13] M. H. Hassoun, *Fundamentals of Artificial Neural Networks*. Massachusetts: MIT Press, Cambridge, 1995.

[14] W. Gao, "Evolutionary Neural Network Based on New Ant Colony Algorithm," presented at the International Symposium on Computational Intelligence and Design (ISCID '08), Wuhan, China, 2008.

[15] A. E. Eiben and J. E. Smith, *Introduction to Evolutionary Computing*. New York: Springer, 2008.

[16] M. Mahdavi, M. Fesanghary, and E. Damangir, "An Improved Harmony Search Algorithm for Solving Optimization Problems," *Applied Mathematics and Computation,* vol. 188, pp. 1567-1579, 2007.

[17] M. G. H. Omran and M. Mahdavi, "Globel-Best Harmony Search," *Applied Mathematics and Computation,* vol. 198, pp. 643-656, 2008.

[18] Q.-K. Pan, P. N. Suganthan, M. F. Tasgetiren, and J. J. Liang, "A self-adaptive global best harmony search algorithm for continuous optimization problems," *Applied Mathematics and Computation,* vol. 216, pp. 830-848, 2010.

[19] K. M. Lane and R. D. Neidinger, "Neural networks from idea to implementation," *ACM Sigapl APL Quote Quad,* vol. 25, pp. 27-37, 1995.

[20] A. T. Chronopoulos and J. Sarangapani, "A distributed discrete-time neural network architecture for pattern allocation and control," in *Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS'02)*, Florida, USA, 2002, pp. 204-211.

[21] T. Jayalakshmi and A. Santhakumaran, "Improved Gradient Descent Back Propagation Neural Networks for Diagnoses of Type II Diabetes Mellitus," *Global Journal of Computer Science and Technology,* vol. 9, 2010.

[22] B. Cetişli and A. Barkana, "Speeding up the scaled conjugate gradient algorithm and its application in neuro-fuzzy classifier training," *Soft Computing,* vol. 14, pp. 365-378, 2010.

[23] A. E. Kostopoulos and T. N. Grapsa, "Self-scaled conjugate gradient training algorithms," *Neurocomputing,* vol. 72, pp. 3000-3019, 2009.

[24] M. S. Bascil and F. Temurtas. (2009, A Study on Hepatitis Disease Diagnosis Using Multilayer Neural Network with Levenberg Marquardt Training Algorithm. *Journal of Medical Systems* [Original Paper].

[25] M. I. Soliman and S. A. Mohamed, "A highly efficient implementation of a backpropagation learning algorithm using matrix ISA," *Journal of Parallel and Distributed Computing,* vol. 68, pp. 949–961, 2008.

[26] T. Kathirvalavakumar and P. Thangavel, "A Modified Backpropagation Training Algorithm for Feedforward Neural Networks," *Neural Processing Letters,* vol. 23, pp. 111-119, 2006.

[27] L. Fausett, *Fundamentals of Neural Networks Architectures, Algorithms, and Applications*. New Jersey: Prentice Hall, 1994.

[28] B. Guijarro-Berdinas, O. Fontenla-Romero, B. Perez-Sanchez, and A. Alonso-Betanzos, "A New Initialization Method for Neural Networks Using Sensitivity Analysis," presented at the International Conference on Mathematical and Statistical Modeling, Ciudad Real, Spain, 2006.

[29] J. Škutova, "Weights Initialization Methods for MLP Neural Networks," *Transactions of the VŠB,* vol. LIV, article No. 1636, pp. 147-152, 2008.

[30] K. P. Ferentinos, "Biological engineering applications of feedforward neural networks designed and parameterized by genetic algorithms," *Neural Networks,* vol. 18, pp. 934-950, 2005.

[31] G. Wei, "Study on Evolutionary Neural Network Based on Ant Colony Optimization," presented at the International Conference on Computational Intelligence and Security Workshops, Harbin, Heilongjiang, China, 2007.

[32] J. Yu, S. Wang, and L. Xi, "Evolving artificial neural networks using an improved PSO and DPSO," *Neurocomputing,* vol. 71, pp. 1054-1060, 2008.

[33] Y. Zhang and L. Wu, "Weights Optimization of Neural Networks via Improved BCO Approach," *Progress In Electromagnetics Research,* vol. 83, pp. 185-198, 2008.

[34] R. E. Dorsey, J. D. Johnson, and W. J. Mayer, "A Genetic Algoirthm for the Training of Feedforward Neural Networks," *Advances in A.pngicial Intelligence in Economics, Finance, and Management* vol. 1, pp. 93-111, 1994.

[35] E. Alba and J. F. Chicano, "Training Neural Networks with GA Hybrid Algorithms," in *Genetic and Evolutionary Computation (GECCO 2004)*. vol. 3102/2004, ed: Springer Berlin / Heidelberg, 2004, pp. 852-863.

[36] D. Kim, H. Kim, and D. Chung, "A Modified Genetic Algorithm for Fast Training Neural Networks," in *Advances in Neural Networks - ISNN 2005*. vol. 3496/2005, ed: Springer Berlin / Heidelberg, 2005, pp. 660-665.

[37] R. Giri, A. Chowdhury, A. Ghosh, S. Das, A. Abraham, and V. Snasel, "A Modified Invasive Weed Optimization Algorithm for training of feed- forward Neural Networks," in *International Conference on Systems Man and Cybernetics*, Istanbul, Turkey, 2010, pp. 3166 - 3173.

[38] D. J. Montana and L. Davis, "Training Feedforward Neural Networks Using Genetic Algorithms," in *Proceedings of the International Joint Conference on Artificial Intelligence*, Detroit, USA, 1989, p. 762.

[39] D. J. Montana, "Neural Network Weight Selection Using Genetic Algorithms," *Intelligent Hybrid Systems,* pp. 85-104., 1995.

[40] R. S. Sexton and R. E. Dorsey, "Reliable classification using neural networks: a genetic algorithm and backpropagation comparison," *Decision Support Systems,* vol. 30, pp. 11-22, 15 December 2000.

[41] M. N. H. Siddique and M. O. Tokhi, "Training neural networks: backpropagation vs. genetic algorithms," in *International Joint Conference on Neural Networks (IJCNN '01)*, Washington, DC 2001, pp. 2673 - 2678.

[42] R. S. Sexton, R. E. Dorsey, and N. A. Sikander, "Simultaneous Optimization of Neural Network Function and Architecture Algorithm," *Decision Support Systems,* vol. 30, pp. 11-22, December 2004 2004.

[43] K. E. Fish, J. D. Johnson, R. E. Dorsey, and J. G. Blodgett, "Using an Artificial Neural Network Trained with a Genetic Algorithm to Model Brand Share " *Journal of Business Research,* vol. 57, pp. 79-85, January 2004 2004.

[44] J. N. D. Gupta and R. S. Sexton, "Comparing backpropagation with a genetic algorithm for neural network training," *Omega, The International Journal of Management Science,* vol. 27, pp. 679-684, 1999.

[45] Z. W. Geem, C.-L. Tseng, and Y. Park, "Harmony Search for Generalized Orienteering Problem: Best Touring in China," in *Advances in Natural Computation*. vol. 3612/2005, ed: Springer Berlin / Heidelberg, 2005, pp. 741-750.

[46] R. S. Sexton, R. E. Dorsey, and J. D. Jhonson, "Towards global optimization of neural networks: A comparison of the genetic algorithm and backpropagation.," *Decision Support Systems,* vol. 22, pp. 171-185, 1998.

[47] E. Fiesler and J. Fulcher, "Neural network classification and formalization," *Computer Standards & Interfaces,* vol. 16, pp. 231-239, July 1994.

[48] I.-S. Oh and C. Y. Suen, "A class-modular feedforward neural network for handwriting recognition," *Pattern Recognition,* vol. 35, pp. 229-244, 2002.

[49] M. T. Ayvas, "Simulation determination of aquifer parameters and zone structures with fuzzy c-means clustering and meta-heuristic harmony search algorithm," *Advances in Water Resources,* vol. 30, pp. 2326-2338, 2007.

[50] Z. W. Geem, "Optimal Cost Design of Water Distribution Networks Using Harmony Search," *Engineering Optimization,* vol. 38, pp. 259-277, 2006.

[51] M. Geethanjali, S. M. R. Slochanal, and R. Bhavani, "PSO trained ANN-based differential protection scheme for power transformers," *Neurocomputing,* vol. 71, pp. 904-918, 2008.

[52] Z. W. Geem and J.-Y. Choi, "Music Composition Using Harmony Search Algorithm," in *Applications of Evolutionary Computing*. vol. 4448/2007, ed: Springer Berlin / Heidelberg, 2007, pp. 593-600.

[53] A. Frank and A. Asuncion. (2010). *UCI Machine Learning Repository, University of California, Irvine, School of Information & Computer Sciences*. Available: http://archive.ics.uci.edu/ml

[54] M. Delgado, M. C. Pegalajar, and M. P. Cuellar, "Memetic Evolutionary Training for Recurrent Neural Networks: An Application to Time-Series Prediction," *Expert Systems,* vol. 23, pp. 99-115, 2006.

[55] D. Randall Wilson and T. R. Martinez, "The general inefficiency of batch training for gradient descent learning," *Neural Networks,* vol. 16, pp. 1429-1451, 2003.

[56] Y. Liu, J. A. Starzyk, and Z. Zhu, "Optimized Approximation Algorithm in Neural Networks Without Overfitting," *IEEE Transactions on Neural Networks,* vol. 19, pp. 983-995, 2008.