EFuNN Ensembles Construction Using a Clustering Method and a Coevolutionary Multi-Objective Genetic Algorithm

L. Minku and T. Ludermir

Federal University of Pernambuco, Informatics Center,
Recife-PE 50732-970, Brazil, P.O.Box 7851,
{flm,tbl}@cin.ufpe.br

Abstract. This paper presents the experiments which where made with the Clustering and Coevolution to Construct Neural Network Ensemble (CONE) approach on two classification problems and two time series prediction problems. This approach was used to create a particular type of Evolving Fuzzy Neural Network (EFuNN) ensemble and optimize its parameters using a Coevolutionary Multi-objective Genetic Algorithm. The results of the experiments reinforce some previous results which have shown that the approach is able to generate EFuNN ensembles with accuracy either better or equal to the accuracy of single EFuNNs generated without using coevolution. Besides, the execution time of CONE to generate EFuNN ensembles is lower than the execution time to produce single EFuNNs without coevolution.

1 Introduction

Several approaches have been developed to optimize parameters of Evolving Connectionist Systems (ECoSs) [1] using evolutionary algorithms, e.g. [2], [3], [4], and [5]. Nevertheless, ensembles of learning machines have been formally and empirically shown to generalize better than single predictors [6]. Instead of utilizing just one neural network to solve a specific problem, an ensemble of neural networks combines a set of neural networks. In order to improve the accuracy of a particular type of ECoSs called Evolving Fuzzy Neural Network (EFuNN), a multi-module classifier called multiEFuNN has been proposed in [7].

However, the construction of ensembles of neural networks is not an easy task [6]. Besides, the choice of the best EFuNN parameters set is also a difficult task and the execution time of evolutionary algorithms to optimize the EFuNN parameters is high. Therefore, a new approach to construct ensembles of neural networks has been proposed in [8] and experiments have been made using a Coevolutionary Genetic Algorithm to generate a particular type of EFuNN ensembles. These experiments have shown that CONE is able to generate EFuNN ensembles with accuracy either better or equal to the accuracy of single EFuNNs generated using a Genetic Algorithm (GA). Moreover, the execution time of CONE to produce EFuNN ensembles is lower than the execution time of the GA.

However, the Coevolutionary GA used in [8] demands the predefinition of some parameters for the fitness function. These parameters have great influence on the results of the evolutionary process. Besides, CONE needs to be evaluated using other coevolutionary algorithms in order to be validated. Thus, a coevolutionary multi-objective GA has been used with CONE in [9] to perform experiments on four classification problems. Nevertheless, CONE needs to be evaluated using not only classification problems, but also time series prediction problems in order to be validated. Thus, this paper presents experiments which have been made with CONE and a coevolutionary multi-objective GA on two classification problems and two time series prediction problems.

This paper is organized as follows: Sect.2 contains an explanation about ECoSs and EFuNNs. Section 3 presents CONE and explains a particular instance of it (i.e. a clustering method, a particular type of EFuNN ensemble and a coevolutionary multi-objective GA which can be used by the approach). Section 4 presents the results of the experiments made with this instance of CONE. Section 5 presents the conclusions and future works.

2 ECoS and EFuNNs

The ECOSs presented in [1] are systems constituted by one or more neural networks. Some of their characteristics are that their learning is on-line, incremental, fast and local [4]. EFuNNs [10] are a class of ECOSs which join the neural networks functional characteristics to the expressive power of fuzzy logic.

The EFuNN learning has some predefined parameters. Using different parameters sets, EFuNNs attain different performances and different weights are learned. The optimal parameters set usually depends on the input and output data presented. Thus, it is important to correctly choose the parameters which define the EFuNN learning according to the data presented.

Some of the predefined parameters of the EFuNN leaning algorithm are the number of membership functions; the initial sensitivity threshold (S) of the nodes (it is also used to determine the initial radius of the receptive field of a node); the error threshold (E); the *m-of-n* value (number of highest activation nodes used in the learning); and the maximum radius of the receptive field ($Mrad$). It is recommended to read [10] to get more details about the EFuNN learning algorithm and its parameters.

3 CONE

This section briefly describes CONE [8]. The general idea of this approach is to construct neural network ensembles using a clustering method to partition the input space in clusters. The training and test patterns are used by the clustering method to create the clusters. After that, the clusters are used to separate the training and the test patterns themselves in various subsets of training and test patterns with empty intersection.Each subset is used to train/test a different population of neural networks, which composes a species that is evolved through

a cooperative coevolutionary algorithm. Thus, each cluster is associated with a training subset, a test subset and a species.

At the end of the evolutionary process, the representatives of each species in the last generation are used to constitute the ensemble. In order to use/test the ensemble, the clusters to which the input test pattern belongs are determined. After that, the outputs of the EFuNNs correspondent to these clusters are calculated and combined using a predefined combining method. Examples of combining methods can be found in [11].

The patterns used by the approach are divided into 3 types: training patterns (used to create clusters and to train the neural networks), test patterns (used to create clusters and to test the neural networks during the evolutionary process), and final test patterns (used to test the neural network ensemble generated at the end of the evolutionary process).

The following sections explain the instance of CONE which has been used in the experiments to produce EFuNN ensembles: Sect.3.1 explains the clustering method used to partition the input space, Sect.3.2 explains the EFuNN ensembles created and Sect.3.3 explains the coevolutionary algorithm used.

3.1 Clustering Method

The clustering method used in the experiments is similar to the Evolving Clustering Method [7]. It is recommended to read [8] to get more details about the clustering method used. The main information about the clustering method for this paper is the *Dthrs* parameter. This is the most important parameter to determine whether a cluster could be updated to accommodate a particular pattern, or if a new cluster would have to be created to accommodate this pattern.

3.2 Creating EFuNN Ensembles

In the experiments performed with CONE, the coevolutionary algorithm was used to optimize the predefined parameters of the EFuNN learning which where cited in Sect.2, and the EFuNN learning algorithm itself was used to train the EFuNNs. A representative of a species was considered the best fit individual of the species. Two combining methods were used to combine the outputs of the EFuNNs that compose the ensemble. One of them is the arithmetic average of the outputs of the EFuNNs to which the pattern presented belongs. The other one is the weighted average of the outputs of the EFuNNs to which the pattern presented belongs. The value used as the weight of a cluster $C_j,\ j = 0, 1, ...N$ is $1/||x_i - Cc_j||$), where x_i is the pattern presented and Cc_j is the cluster center. If a pattern does not belong to any cluster, the output of the ensemble is the output of the EFuNN correspondent to the cluster whose center is the nearest center to the pattern.

3.3 Coevolutionary Algorithm

This section describes the coevolutionary multi-objective GA which was used in the experiments. It is recommendable to read [9] to get more details about it.

The coevolutionary multi-objective GA used in the experiments has a binary representation of the EFuNN parameters to be optimized, bitwise bit-flipping mutation, one-point crossover, generational survivor selection, and the learning algorithm of the neural networks being optimized is used with the training patterns right before the calculation of the objective values.

The initial population of each species is composed by individuals created randomly choosing values for each of the EFuNN parameters to be optimized. In this population, the objectives vector of an individual i is:

$$[RMSE\_Obj_i = RMSE_i,\ SIZE\_Obj_i = size_i]\ , \tag{1}$$

where $RMSE_i$ is the Root Mean Squared Error (RMSE) obtained testing the EFuNN correspondent to the individual i using the test subset correspondent to its species, and $size_i$ is the size of this EFuNN. Thus, the objectives are calculated without considering the individuals of the other species. The size component of the objectives vector is used to penalize the size of the EFuNNs and reduce the execution time of the evolutionary algorithm, as suggested in [5].

In all generations after the initial one, the objectives of an individual i are calculated using not only the output error and the size of the EFuNN correspondent to i, but also the output error and size of the EFuNNs correspondent to the representatives of the other species in the previous generation. Thus, the objectives vector of an individual i is $[RMSE\_Obj_i,\ SIZE\_Obj_i]$, where:

$$RMSE\_Obj_i = \sqrt{\frac{SSE_i + repr\_sse}{total\_test\_patterns\_number}}\ \text{and} \tag{2}$$

$$SIZE\_Obj_i = size_i + repr\_size\ . \tag{3}$$

In these equations, SSE_i is the Sum of Squared Error (SSE) obtained testing the EFuNN correspondent to the individual i with the test subset correspondent to its species; $size_i$ is the size of this EFuNN; $repr\_sse$ is the sum of the SSEs and $repr\_size$ is the sum of the sizes of the EFuNNs correspondent to the representatives of all other species in the previous generation; and $total\_test\_patterns\_number$ is the total number of test patterns, including the patterns of all species.

An individual i dominates an individual j if $(RMSE_i \leq RMSE_j)$ and $(SIZE_i \leq SIZE_j)$ and $(RMSE_i < RMSE_j$ or $SIZE_i < SIZE_j)$.

After the calculation of the objective values, the rank of each individual is calculated. As it is done in [12], the rank of an individual i of the population p in the generation g is the number of individuals $j \neq i,\ j \in p$ which dominate the individual i in the generation g. In this way, a pareto optimal individual (an individual which is not dominated by any other individual of the population) has always rank equal to 0.

The best individual of a population is the individual which has the lowest rank. When more than 1 individual has the same rank, the best between them is the one which has the lowest SSE obtained testing the EFuNN correspondent to it with the test subset correspondent to its species.

The parents selection is made using the roulette wheel method and is proportional to the value determined by be following equation:

$$Prob_{i,p,g} = \frac{max\_rank_{p,g} - rank_{i,p,g}}{\sum_{j=0}^{pop\_size_p - 1}(max\_rank_{p,g} - rank_{j,p,g})} \ , \tag{4}$$

where $max\_rank_{p,g}$ is highest rank of the population p in the generation g, $rank_{i,p,g}$ is the rank of the individual i of the population p in the generation g, and pop_size_p is the size of the population p.

4 Experiments

This section explains the experiments which have been made with the instance of CONE described in Sects.3.1, 3.2 and 3.3. The experiments have utilized two classification databases (Card and Diabetes) [13] and two time series prediction problems (Mackey-Glass (MG) [14] and Gas Furnace (GF) [15]).

Section 4.1 shows the parameters which were used in the experiments and Sect.4.2 presents the results of the experiments.

4.1 Parameters and Executions

The parameters utilized in the experiments were the same as the parameters used in [9], except the $Dthrs$. The $Dthrs$ was empirically determined for each database and it was 0.40 for Card database, 0.25 for Diabetes database, 0.20 for Mackey-Glass time series and 3.00 for Gas Furnace time series. The EFuNN parameters optimized during the coevolutionary process were also the same as the optimized in [9]: m-of-n, E, $Mrad$, S and membership functions number.

Three different partitions of the training+test and final_test data sets of the classification problems were used and three different time series were used to compose 3 partitions of the training+test and final_test data sets for the time series problems:

- Mackey-Glass:
  $w(1) = [x(t-12)x(t-8)x(t-4)x(t); y(t+4)]$
  $w(2) = [x(t-18)x(t-12)x(t-6)x(t); y(t+6)]$
  $w(3) = [x(t-24)x(t-16)x(t-8)x(t); y(t+8)]$
- Gas Furnace:
  $w(1) = [y(t-1)y(t-2)x(t-1)x(t-2); y(t)]$
  $w(2) = [y(t-1)y(t-3)x(t-1)x(t-3); y(t)]$
  $w(3) = [y(t-2)y(t-3)x(t-2)x(t-3); y(t)]$

Ten executions with different random seeds were performed for each partition, thus totalizing 30 executions for each database. Executions with the above combinations of parameters were also made using a multi-objective GA to generate single EFuNNs. The multi-objective GA utilized was the same as the algorithm presented in Sect.3.3, but using the objectives (1) for all generations and just

one species. In this way, 30 executions of the multi-objective GA were made for each database.

The objective of the executions explained above was to compare:

- EFuNN ensembles generated using CONE with weighted average combining method (weighted EFuNN ensembles – WEns);
- EFuNN ensembles generated using CONE with arithmetic average combining method (arithmetic EFuNN ensembles – AEns);
- Single EFuNNs generated using multi-objective GA (Sing).

The characteristics compared were the execution times of the evolutionary approaches, and the output classification errors/sum of squared errors(SSEs) of the EFuNN ensembles and of the single EFuNNs generated.

4.2 Results

In this section, the classification errors and the SSEs are those obtained using the final_test patterns set to test the single EFuNNs or the EFuNN ensembles generated after the evolutionary processes. The classification errors are used for the classification problems and the SSEs are used for the time series problems.

Table 1 shows the classification error/SSE and execution time averages, standard deviations, minimal and maximum values, considering the 30 executions for each database. Table 2 shows the statistics of the T student tests [16] performed to compare the classification errors/SSEs and the execution times. As it can be seen, the classification error/SSE averages of the ensembles created for all databases were considered statistically equal to the classification error/SSE averages of the single EFuNNs. The classification error averages of the weighted EFuNN ensembles were also considered statistically equal to the classification error averages of the arithmetic EFuNN ensembles, for the classification problems. However, for the time series problems, the SSE averages of the weighted EFuNN ensembles were statistically lower than the SSE averages of the arithmetic EFuNN ensembles.

For all databases, the execution time average among all 30 executions of CONE to generate EFuNN ensembles was statistically lower than the execution time average among all 30 executions of the multi-objective GA to generate single EFuNNs. Table 2 shows the statistics of the T student tests made to prove this analysis.

The execution time of the CONE to generate EFuNN ensembles was lower than the multi-objective GA execution time to generate single EFuNNs probably because in the optimization process of a single EFuNN, for each pattern presented to train/test the EFuNN, the activation levels of all rule nodes of the EFuNN have to be calculated. When an EFuNN ensemble is being created, just the activation levels of the rule nodes of the correspondent EFuNNs have to be calculated. A single EFuNN is usually higher than each EFuNN which compose an ensemble because a single EFuNN has to accommodate all training patterns and a component of an ensemble has to accommodate only the patterns correspondent to a particular cluster of the input space.

Table 1. Measures related to the class error/SSEs and execution times

		Class errors		SSEs		Execution times			
		Card	Diabetes	MG	GF	Card	Diabetes	MG	GF
WEns	Av	0.1611	0.2717	0.0621	182.2633	3122.5333s	804.6s	155s	70.8333s
	SD	0.0276	0.0304	0.0188	118.8774	652.2496s	174.3222s	45.8333s	16.7396s
	Min	0.1214	0.2083	0.0323	79.9432	2063s	500s	76s	49s
	Max	0.2370	0.3281	0.1262	429.2530	4228s	1174s	232s	106s
AEns	Av	0.1611	0.2744	0.0645	184.3665				
	SD	0.0281	0.0363	0.0184	119.1835		Equal to WEns		
	Min	0.1214	0.1875	0.0357	81.7392				
	Max	0.2370	0.3594	0.1274	432.929				
Sing	Av	0.1694	0.2722	0.0536	171.2507	6717.9333s	2200.8333s	746.8333s	184.8s
	SD	0.0294	0.0289	0.0270	124.6976	1285.7797s	467.8189s	208.9156s	55.9411s
	Min	0.1272	0.2187	0.0194	70.7992	4274s	1172s	297s	101s
	Max	0.2312	0.3333	0.1028	391.727	9536s	3396s	1112s	275s

Table 2. T Student test statistics comparing the class error/SSE averages and the execution time averages, using level of significance equal to 0.05

		Card	Diabetes	MG	GF
Class error/SSE averages comparisons	WEns x AEns	-0.0001	-1.0523	-7.8697	-5.3195
	WEns x Sing	-1.5590	-0.0893	1.4294	1.3073
	AEns x Sing	-1.5548	0.3334	1.8200	1.5590
Execution time averages comparisons	WEns x Sing	-13.2646	-15.5206	-15.1242	-12.4373
	AEns x Sing	Equal to WEns x Sing			

5 Conclusions

This paper presents experiments which have been made with CONE using a coevolutionary multi-objective GA. The experiments have used two classification problems and two time series prediction problems.

The experimental results have shown that the EFuNN ensembles construction using CONE has a lower execution time than the single EFuNNs construction using a multi-objective GA. The standard deviations of the execution times are also lower for CONE. Even so, the EFuNN ensembles generalization abilities are statistically equal to the single EFuNNs ones. These results contribute to the validation of CONE, reinforcing the results presented in [8] and [9], which have shown that CONE is able to produce EFuNN ensembles with either equal or better generalization using a lower execution time than similar non-coevolutionary algorithms to produce single EFuNNs.

Future works include the use of other clustering methods and coevolutionary algorithms to create neural network ensembles using CONE.

Acknowledgment

This work was supported by CNPq – Brazil.

References

1. Kasabov, N.: Evolving Connectionist Systems. Springer, Great Britain (2003)
2. Watts, M., Kasabov, N.: Dynamic optimisation of evolving connectionist system training parameters by pseudo-evolution strategy. In: CEC'2001. Volume 2., Seoul (2001) 1335–1342
3. Watts, M., Kasabov, N.: Evolutionary optimisation of evolving connectionist systems. In: CEC'2002. Volume 1., Honolulu, Hawaii, IEEE Press (2002) 606–610
4. Kasabov, N., Song, Q., Nishikawa, I.: Evolutionary computation for dynamic parameter optimization of evolving connectionist systems for on-line prediction of time series with changing dynamics. In: IJCNN'2003. Volume 1., Oregon (2003) 438–443
5. Minku, F.L., Ludermir, T.B.: Evolutionary strategies and genetic algorithms for dynamic parameter optimization of evolving fuzzy neural networks. In: CEC'2005. Volume 3., Edinburgh, Scotland (2005) 1951–1958
6. Chandra, A., Yao, X.: Ensemble learning using multi-objective evolutionary algorithms. Journal of Mathematical Modelling and Algorithms (1) (2006)
7. Kasabov, N.: Ensembles of efunns: An architecture for a multimodule classifier. In: Proceedings of the International Conference on Fuzzy Systems. Volume 3., Australia (2001) 1573–1576
8. Minku, F.L., Ludermir, T.B.: EFuNNs ensembles construction using a clustering method and a coevolutionary genetic algorithm (to appear). In: CEC'2006, Vancoucer, Canada (2006)
9. Minku, F.L., Ludermir, T.B.: EFuNN ensembles construction using CONE with multi-objective GA (to appear). In: SBRN'2006, Ribeirao Preto, Brazil (2006)
10. Kasabov, N.: Evolving fuzzy neural networks for supervised/unsupervised on-line, knowledge-based learning. IEEE Transactions on Systems, Man and Cybernetics **31**(6) (2001) 902–918
11. Dietterich, T.G.: Machine-learning research: Four current directions. The AI Magazine **18**(4) (1998) 97–136
12. Fonseca, C.M., Fleming, P.J.: Multi-objective optimization and multiple constraint handling with evolutionary algorithms - part I: A unified formulation. IEEE Transactions on Systems, Man and Cybernetics - Part A **28**(1) (1998) 26–37
13. Prechelt, L.: PROBEN1 - a set of neural network benchmark problems and benchmarking rules. Technical Report 21/94, Fakultt fr Informatik, Universitt Karlsruhe, Karlsruhe, Germany (1994)
14. Mackey, M.C., Glass, L.: Oscillation and chaos in physiological control systems. Science **197** (1977) 287–289
15. Box, G.E.P., Jenkins, G.M.: Time Series Analysis, Forecasting and Control. Holden Day, San Francisco (1970)
16. Witten, I.H., Frank, E.: Data Mining - Pratical Machine Learning Tools and Techniques with Java Implementations. Morgan Kaufmann Publishers, San Francisco (2000)