EFuNN Ensembles Construction Using CONE with Multi-objective GA

L. Minku, 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 four classification benchmark databases. 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) [8]. Among them, [14] describes a try to optimize some ECoSs parameters in an on-line manner, [15] presents a method to optimize the parameters and the order of presentation of the training patterns in an off-line manner, [9] and [10] present successfully methods to optimize ECoSs parameters in an on-line manner. All these methods use evolutionary algorithms to make the optimization of the ECoSs parameters.

Nevertheless, ensembles of learning machines have been formally and empirically shown to generalize better than single predictors [1]. 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 [6].

However, the construction of ensembles of neural networks is not an easy task [1]. In particular, it is important to observe that the components of an ensemble should have errors at least somewhat not correlated and each component of an ensemble should have small error rates in order to constitute a successful ensemble [2]. 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 [11] 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 is lower than the execution time of GA.

However, the Coevolutionary GA used in [11] 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 new coevolutionary multi-objective GA has been proposed to be used with CONE. This paper presents the experiments which have been made with CONE on four benchmark databases using this coevolutionary multi-objective GA. As in [11], the experiments were made to continue the validation process of CONE and to improve the accuracy of systems based on EFuNNs.

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 [8] are systems constituted by one or more neural networks. Some of their characteristics are that their learning is on-line, incremental, fast and local [9]. EFuNNs [7] 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 [7] to get more details about the EFuNN learning algorithm and its parameters.

3. CONE

This section describes CONE [11]. 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, as Fig.1 shows. 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.

The coevolutionary algorithm can be used to optimize the architecture of the neural networks or, for example, the weights of the connections. Thus, it is possible to use the coevolutionary algorithm both to train and to optimize the architecture of the neural networks. However, it is also possible to use the specific learning algorithm of the neural networks to train them and the coevolutionary algorithm just to optimize their architectures.

At the end of the evolutionary process, an ensemble is created using a representative of each species in the last generation, as shown by Fig.1. Figure 2 illustrates a neural network ensemble created using CONE. It is important to observe that, although each species has just one representative, a pattern could belong to more than one cluster. 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.

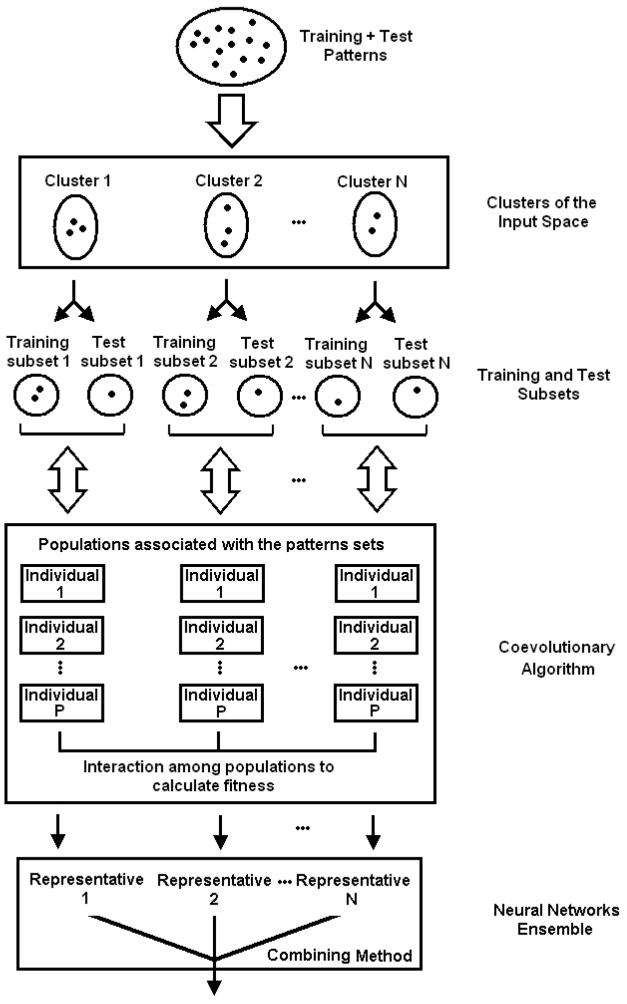

Figure 1. CONE

Examples of combining methods can be found in [2].

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 evolutionary process is cooperative because the evaluation of an individual of a species is made using a representative individual of each one of the other species to constitute a neural network ensemble. The representative of a species can be, for example, its best individual. However, the interaction among individuals of different species occurs only in their evaluation. So, there is no matching between individuals of different species.

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

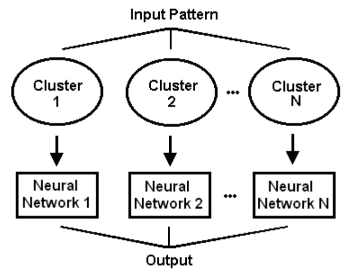

Figure 2. Neural network ensemble

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 [6]. The following algorithm presents it ($NumEx$ is the number of patterns and $Dthr$ is a predefined distance threshold):

1. Create the first cluster C_0 by simply taking the position of the first pattern as the first cluster center Cc_0 and setting a value 0 for its cluster radius Ru_0.
2. For each input pattern x_i from $i = 1$ to $NumEx - 1$ do:
   (a) Determine the distance between x_i and all N cluster centers Cc_j already created:
   $$D_{ij} = ||x_i - Cc_j||,\ j = 0, 1, ..., N - 1$$
   (b) If there is a distance value $D_{ij} \leq Ru_j$, it means that x_i belongs to a cluster C_m with the minimum distance $D_{im} = ||x_i - Cc_m|| = min(||x_i - Cc_j||)$, subject to the restriction $D_{ij} \leq Ru_j,\ j = 0, 1, ..., N - 1$. In this case, neither a new cluster is created nor an existing cluster is updated.
   (c) Else
      i. Find the cluster C_α with the minimum distance $D_{i\alpha} = ||x_i - Cc_\alpha|| = min(||x_i - Cc_j||),\ j = 0, 1, ..., N - 1$.
      ii. If $D_{i\alpha} > Dthr$, create a new cluster, in the same way as described in the step 1.
      iii. Else update C_α: increment the number of patterns accommodated by C_α ($NExs_\alpha = NExs_\alpha + 1$); update Cc_α ($Cc_\alpha = Cc_\alpha + (x_i - Cc_\alpha)/NExs_\alpha$) and make R_α be the maximum between the following values: 1. the distance between the old Cc_α and the new Cc_α plus the old Ru_α and 2. the distance between x_i and the new center Cc_α

In this paper, the distance between two vectors x and y denotes the General Euclidean Distance, defined as follows:

$$||x - y|| = \sqrt{\frac{\sum_0^{size-1}(x_i - y_i)^2}{size}}$$

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

The experiments made with CONE have used a coevolutionary multi-objective GA. It is recommendable to read [4] for an explanation about evolutionary algorithms and [12] for an example of a coevolutionary approach.

The coevolutionary multi-objective GA used has a binary representation of the EFuNN parameters to be optimized, bitwise bit-flipping mutation, one-point crossover, and generational survivor selection.

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:

$$[RMSE\_Obj_i = RMSE_i,\ SIZE\_Obj_i = size_i]\ , \quad (1)$$

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 [10].

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 this individual, 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:

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

$$SIZE\_Obj_i = size_i + repr\_size \ . \quad (3)$$

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 [5], 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:

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

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.

Each species is evolved in a separate manner and there is an interaction among the species only to calculate the objective values, according to CONE. The following algorithm is the algorithm used to evolve a specific species:

1. Create the initial population.
2. Repeat until a maximum number of generations is attained:
   (a) Apply the EFuNN learning to each EFuNN of the population using the training subset correspondent to this species and the parameters codified by the genotype of the individuals, and test them using the test subset.
   (b) Determine the objective values and the rank of the individuals.
   (c) Make parent selection according to (4).
   (d) Apply crossover and mutation with probabilities *Pc* and *Pm*, respectively, to generate new individuals. The new individuals do not inherit the rule nodes of their parents.
   (e) Apply generational survivor selection.

4. Experiments

This section explains the experiments which have been made with the instance of CONE presented in Sects.3.1, 3.2 and 3.3. The experiments have utilized four benchmark databases: Iris Plant, Wine, Glass and Cancer. These databases were obtained from the UCI Machine Learning Repository [3] and from Proben1 [13]. Section 4.1 shows the parameters used and Sect.4.2 presents the results of the experiments.

4.1. Parameters and executions

The parameters utilized in the experiments were chosen according to the experiments made in [11]. The EFuNN parameters optimized during the coevolutionary process and their intervals of allowed values were: m-of-n ($[1, 15] \in Z$), E ($[0.01, 0.6] \in R$), $Mrad$ ($[0.01, 0.8] \in R$), S ($[0.4, 0.99] \in R$) and membership functions number ($[2, 8] \in Z$). The $Dthrs$ parameter of the clustering method was empirically determined and it is 0.40 for Iris database, 50 for Wine database, 0.20 for Glass database and 0.37 for Cancer database. It is important to observe that the coice of the $Dthrs$ parameter depends on the database and it usually is greater if the attributes of the database can assume greater values. The parameters of the coevolutionary algorithm were: population size = 12, *Pm* = 2%, *Pc* = 70%, W_{rmse} = 0.1, and stop criterium = 50 generations.

Three different partitions of the training+test and final_test data sets were also used, thus totalizing 3 combinations of configurations. All training+test partitions were composed by 75% of the patterns, all final_test partitions were composed by 25% of the patterns of the database, 66% of the training+test patterns were used to train the EFuNNs, and 33% were used to test them during the evolutionary process. Ten executions with different random seeds were performed for each combination, 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 of the EFuNN ensembles and of the single EFuNNs generated.

4.2. Results

In this section, the classification errors are those obtained using the final_test patterns set to test the single EFuNNs or the EFuNN ensembles generated after the evolutionary processes. Table 1 shows the classification error averages, standard deviations, minimal and maximum values, considering the 30 executions for each database.

It can be observed that for the Iris, Wine and Glass databases the classification error averages of the ensembles were lower than the classification error averages of the single EFuNNs. For the Cancer database, the classification error averages of the ensembles were greater than the classification error average of the single EFuNNs. Nevertheless, the classification error averages of the ensembles created for all databases were considered statistically equal to the classification error averages of the single EFuNNs. The classification error averages of the weighted EFuNN ensembles were also always considered statistically equal to the classification error averages of the arithmetic EFuNN ensembles.

Table 2 shows the statistics of the T student tests [16] performed to prove the analysis made with the classification errors. The signals "=" indicate that the classification errors of all 30 executions of the compared approaches were equal.

Table 1. Measures related to the classification errors

		Iris	Wine	Glass	Cancer
WEns	Av	0.0521	0.0259	0.3107	0.0402
	SD	0.0378	0.0292	0.0547	0.0134
	Min	0.0256	0	0.2264	0.0172
	Max	0.1282	0.1111	0.4151	0.0805
AEns	Av	0.0521	0.0259	0.3132	0.0439
	SD	0.0378	0.0292	0.0562	0.0156
	Min	0.0256	0	0.2075	0.0115
	Max	0.1282	0.1111	0.4151	0.0805
Sing	Av	0.0598	0.0274	0.3302	0.0379
	SD	0.0304	0.0162	0.0924	0.0107
	Min	0	0	0.1509	0.0172
	Max	0.1282	0.0444	0.5471	0.0575

Table 2. T Student test statistics comparing the classification error averages using level of significance equal to 0.05

	Iris	Wine	Glass	Cancer
WEns x AEns	=	=	-1.2782	-2.0266
WEns x Sing	-1.2477	-0.2352	-1.1802	0.7793
AEns x Sing	-1.2477	-0.2352	-1.0471	1.9800

Table 3 shows the execution time averages, standard deviations, minimal and maximum values, considering the 30 executions for each database. 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 4 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.

It is also interesting to observe that the standard de-

Table 3. Measures related to the execution times in seconds

		Iris	Wine	Glass	Cancer
Ens	Av	34.033	30.433	59.067	421.733
	SD	2.684	3.126	11.965	170.659
	Min	29	25	38	196
	Max	43	36	77	884
Sing	Av	134.267	99.467	101.733	984.033
	SD	39.505	23.263	24.189	248.590
	Min	87	66	60	527
	Max	226	146	154	1701

Table 4. T Student test statistics comparing the execution time averages using level of significance equal to 0.05

	Iris	Wine	Glass	Cancer
Ens x Sing	-13.6713	-15.8802	-11.1407	-10.2514

viations of the execution times of the CONE to produce EFuNN ensembles were lower than the standard deviations of the execution times of the multi-objective GA to produce single EFuNNs for all databases, as table 3 shows.

5. Conclusions

This paper proposes a coevolutionary multi-objective GA which can be used with CONE to create EFuNN ensembles and it presents experiments on four benchmark problems using a particular instance of CONE that uses this coevolutionary multi-objective GA.

The experimental results have shown that the EFuNN ensembles construction using the proposed instance of 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 equal to the single EFuNNs ones. These results contribute to the validation of CONE, reinforcing the results presented in [11], 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 and the comparison of CONE with other approaches.

Acknowledgment

This work was supported by CNPq – Brazil.

References

[1] A. Chandra and X. Yao. Ensemble learning using multi-objective evolutionary algorithms. *Journal of Mathematical Modelling and Algorithms*, (1), 2006.

[2] T. G. Dietterich. Machine-learning research: Four current directions. *The AI Magazine*, 18(4):97–136, 1998.

[3] C. B. D.J. Newman, S. Hettich and C. Merz. UCI repository of machine learning databases, 1998.

[4] A. E. Eiben and J. E. Smith. *Introduction to Evolutionary Computing*. Springer, Berlin, 2003.

[5] C. M. Fonseca and P. J. Fleming. Multi-objective optimization and multiple constraint handling with evolutionary algorithms - part I: A unified formulation. *IEEE Trans. on Systems, Man and Cybernetics - Part A*, 28(1):26–37, 1998.

[6] N. Kasabov. Ensembles of efunns: An architecture for a multimodule classifier. In *Proceedings of the FUZZ-IEEE*, volume 3, pages 1573–1576, Australia, 2001.

[7] N. Kasabov. Evolving fuzzy neural networks for supervised/unsupervised on-line, knowledge-based learning. *IEEE Trans. on Sys., Man and Cyb.*, 31(6):902–918, Dec. 2001.

[8] N. Kasabov. *Evolving Connectionist Systems*. Springer, Great Britain, 2003.

[9] N. Kasabov, Q. Song, and I. Nishikawa. 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, pages 438–443, Oregon, Jul. 2003.

[10] F. L. Minku and T. B. Ludermir. Evolutionary strategies and genetic algorithms for dynamic parameter optimization of evolving fuzzy neural networks. In *CEC'2005*, volume 3, pages 1951–1958, Edinburgh, Scotland, Sep. 2005.

[11] F. L. Minku and T. B. Ludermir. EFuNNs ensembles construction using a clustering method and a coevolutionary genetic algorithm (to appear). In *CEC'2006*, Canada, 2006.

[12] M. A. Potter and K. A. De Jong. Cooperative coevolution: An architeture for evolving coadapted subcomponents. *Evolutionary Computation*, 8(1):1–29, 2000.

[13] L. Prechelt. PROBEN1 - a set of neural network benchmark problems and benchmarking rules. Technical Report 21/94, Fakultt fr Informatik, Universitt Karlsruhe, Karlsruhe, Germany, Sep. 1994.

[14] M. Watts and N. Kasabov. Dynamic optimisation of evolving connectionist system training parameters by pseudo-evolution strategy. In *CEC'2001*, volume 2, pages 1335–1342, Seoul, May 2001.

[15] M. Watts and N. Kasabov. Evolutionary optimisation of evolving connectionist systems. In *CEC'2002*, volume 1, pages 606–610, Honolulu, Hawaii, May 2002. IEEE Press.

[16] I. H. Witten and E. Frank. *Data Mining - Pratical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann Publishers, San Francisco, 2000.