Simultaneous Evolution of Neural Network Topologies and Weights for Classification and Regression[*]

Miguel Rocha[1], Paulo Cortez[2], and José Neves[1]

[1] Dep. Informática, Universidade do Minho, 4710-057 Braga, Portugal
{mrocha, jneves}@di.uminho.pt
[2] Dep. Sistemas de Informação, Univ. do Minho, 4800-058 Guimarães, Portugal
pcortez@dsi.uminho.pt
http://www.dsi.uminho.pt/~pcortez

Abstract. *Artificial Neural Networks (ANNs)* are important *Data Mining (DM)* techniques. Yet, the search for the optimal *ANN* is a challenging task: the architecture should learn the input-output mapping without overfitting the data and training algorithms tend to get trapped into local minima. Under this scenario, the use of *Evolutionary Computation (EC)* is a promising alternative for *ANN* design and training. Moreover, since *EC* methods keep a pool of solutions, an *ensemble* can be build by combining the best *ANNs*. This work presents a novel algorithm for the optimization of *ANNs*, using a direct representation, a structural mutation operator and Lamarckian evolution. Sixteen real-world classification/regression tasks were used to test this strategy with single and ensemble based versions. Competitive results were achieved when compared with a heuristic model selection and other *DM* algorithms.

Keywords: Supervised Learning, Multilayer Perceptrons, Evolutionary Algorithms, Ensembles.

1 Introduction

Artificial Neural Networks (ANNs) denote a set of connectionist models inspired in the behavior of the human brain. In particular, the *Multilayer Perceptron (MLP)* is the most popular *ANN* architecture, where *neurons* are grouped in *layers* and only *forward connections* exist [3]. This provides a powerful base-learner, with advantages such as nonlinear mapping and noise tolerance, increasingly used in the *Data Mining (DM)* and *Machine Learning (ML)* fields due to its good behavior in terms of predictive knowledge [8].

The interest in *MLPs* was stimulated by the advent of the *Backpropagation* algorithm in 1986 and since then, several fast variants have been proposed (e.g., *RPROP*) [9]. Yet, these training algorithms minimize an error function by tuning the modifiable parameters of a fixed architecture, which needs to be set a

[*] This work was supported by the ALGORITMI research center and the *FCT* project POSI/EIA/59899/2004.

J. Cabestany, A. Prieto, and D.F. Sandoval (Eds.): IWANN 2005, LNCS 3512, pp. 59–66, 2005.

priori. The *MLP* performance will be sensitive to this choice: a small network will provide limited learning capabilities, while a large one will induce generalization loss (i.e., overfitting). Thus, the correct design of the *MLP* topology is a complex and crucial task, commonly addressed by trial-and-error procedures (e.g. exploring different number of hidden nodes), in a *blind* search strategy, which only goes through a small set of possible configurations. More elaborated methods have also been proposed, such as *pruning* [14] and *constructive* [5] algorithms, although these perform *hill-climbing*, being prone to local minima. In addition, the gradient-based procedures used for the *MLP* training are not free from getting trapped into local minima when the error surface is rugged, being also sensitive to parameter settings and to the network initial weights.

An alternative is to optimize both the structure and weights by using *Evolutionary Computation (EC)*, which performs a global multi-point (or *beam*) search, quickly locating areas of high quality, even when the search space is very complex. The combination of *EC* and *ANNs*, called *Evolutionary Neural Networks (ENNs)*, is a suitable candidate for topology design, due to the error surface features [16]: the number of nodes/connections is unbounded; the mapping from the structure to its performance is indirect; changes are discrete; and similar topologies may present different performances. Moreover, since *EC* performs a global search, it is expected to overcome local minima and reach the optimal set of weights. *Ensembles* are another promising *DM/ML* research field, where several models are combined to produce an answer [2]. Often, it is possible to build ensembles that are better than individual learners. One interesting way to build *ANN* ensembles is based on the use of *heterogeneous topologies*, where a family of *MLPs* with distinct structures (and therefore complexities) are combined [11]. Since *ENNs* use a population of different neural structures, this strategy can be easily adapted to *ENNs* with no computational effort increase.

In this work, a novel *ENN* is presented for the simultaneous optimization of *MLPs*, where a direct representation is used. New topologies are achieved by applying a structural mutation, which adds or deletes connections or weights. On the other hand, connection weights are optimized through Lamarckian evolution that uses a random mutation and a local learning algorithm (RPROP). This technique will be tested in classification and regression problems, using both single *ANN* and ensemble based models. Then, the results will be compared with a heuristic *ANN* selection procedure, as well with other *DM/ML* methods.

The paper is organized as follows. First, a description is given on the datasets used (Section 2.1). Then, the neural and evolutionary models are presented (Sections 2.2 and 2.3). In Section 3 the experiments performed are described and the results analyzed. Finally, closing conclusions are drawn in Section 4.

2 Materials and Methods

2.1 Classification and Regression Datasets

This work endorses two important *DM/ML* problems: *classification* and *regression* tasks. The former requires a correct association between input attributes

Table 1. A summary of the data sets used

Task	**Inputs**			**Examples**	**Classes**
	Num.	**Bin.**	**Nom.**		
Balance	4	0	0	625	3
Bupa	6	0	0	345	2
Car	0	0	6	1728	4
Cmc	5	3	1	1473	3
Dermatology	34	0	0	366	6
Ionosphere	34	0	0	351	2
Sonar	60	0	0	104	2
Yeast	7	1	0	1484	10
Abalone	7	0	1	4177	$\Re$
Auto-mpg	5	0	2	398	$\Re$
Autos	17	3	5	205	$\Re$
Breast-cancer	1	4	4	286	$\Re$
Heart-disease	6	3	4	303	$\Re$
Housing	12	1	0	506	$\Re$
Servo	2	0	2	167	$\Re$
WPBC	32	0	0	194	$\Re$

and a class label (e.g., classifying cells for cancer diagnosis). The latter deals with a functional approximation between n-dimensional input vectors and m-dimensional output ones (e.g., stock market prediction).

Eight classification and eight regression datasets were selected from the UCI *ML* repository [13]. The main features are listed in Table 1, namely: the number of numeric (**Num.**), binary (**Bin.**) and nominal (**Nom.**, i.e. discrete with 3 or more labels) input attributes, as well as the number of examples and classes. The regression tasks are identified by the symbol $\Re$ (last eight rows).

2.2 Neural Networks

The *MLPs* used in this study make use of biases, sigmoid activation functions and one hidden layer with a variable number of nodes. A different approach was followed for the regression tasks, since outputs may lie out of the logistic output range ($[0, 1]$). Hence, shortcut connections and *linear* functions were applied on the output neuron(s), to scale the range of the outputs (Fig. 1).

Before feeding the *MLPs*, the data was preprocessed with a *1-of-C* encoding, one binary variable per class, applied to the nominal attributes and all inputs were rescaled within the range $[-1, 1]$. For example, the **safety** attribute from the task **car** was encoded as: *low* → (1 -1 -1), *med* → (-1 1 -1) and *high*→ (-1 -1 1). Regarding the outputs, the discrete variables were normalized within the range $[0, 1]$ (using also a *1-of-C* encoding for the nominal attributes). Therefore, the predicted class is given by the nearest class value to the node's output, if one single node is used (binary variable), otherwise the node with the highest output value is considered. On the other hand, regression problems will be modeled by one real-valued output, which directly represents the *dependent* target variable.

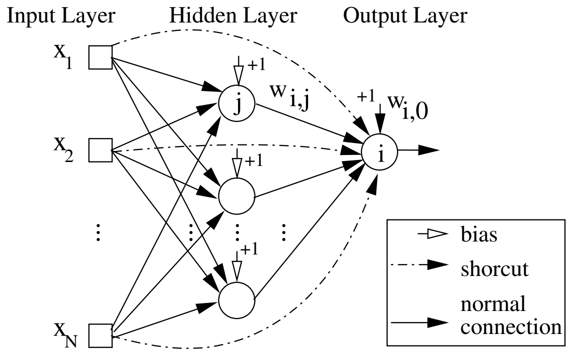


Fig. 1. A fully connected *Multilayer Perceptron* with one output neuron, bias and shortcuts

Two distinct accuracy measures were adopted: the *Percentage of Correctly Classified Examples (PCCE)*, used in classification tasks; and the *Normalized Root Mean Squared Error (NRMSE)*, applied in the regression ones. These measures are given by the equations:

$$\begin{array}{l} PCCE = \sum_{i=1}^{E} 1 \text{ , } if(T_i = P_i)/E \times 100\,(\%) \\ RMSE = \sqrt{\sum_{i=1}^{E} (T_i - P_i)^2/E} \\ NRMSE = \frac{RMSE}{\sum_{i=1}^{E} T_i/E} \times 100\,(\%) \end{array} \qquad (1)$$

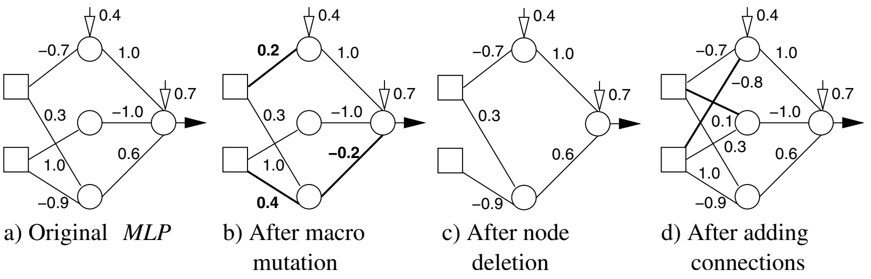
where E denotes the number of examples; P_i, T_i the predicted and target values for the i-th example.

In order to provide a basis for comparison with the *ENN*, an *Heuristic* approach (*HNN*) to model selection was defined by a simple trial-and-error procedure, where fully connected *MLPs*, with a number of hidden nodes ranging from 0 to 20, are trained. For each *MLP*, the initial weights were randomly set within the range $[-1, 1]$. Next, the *RPROP* algorithm [9] was selected for training, due to its faster convergence and stability, being stopped after a maximum of 500 epochs or when the error slope was approaching zero. Then, the topology with the lowest validation error (computed over non training data) is selected. The trained *MLPs* will also be used to build an *Ensemble* (*HNNE*), where the output is given by the average over all 21 *MLPs*.

2.3 Evolutionary Neural Network

In the past, evolutionary approaches have been proposed for training connection weights, optimizing neural topologies and evolving both architectures and weights [16]. Yet, in order to train an *ANN*, an a priori architecture needs to be set. On the other hand, evolving neural structures without weight information will make harder the fitness evaluation due to the noisy fitness evaluation problem: different random initial weights may produce distinct performances. Hence, it seems natural to use the global search advantages of the *EC* to simultaneous evolve topologies and weights [17].

In the present work, a *Simultaneous Evolutionary Neural Network (SENN)* algorithm with a *direct* representation is embraced, where the genotype is the whole *MLP*. The population size contains P individuals and the initial population is created by choosing structures with a random number of hidden nodes (between 0 and H). Then, each possible connection is set with a probability of 50%. Next, the connection weights are randomly initialized within the range $[-1.0; 1.0]$. Regarding the genetic recombination, the crossover operator was discarded since previous experiments [10] revealed no gain in its use, probably due to the *permutation* problem; i.e., several genomes may encode the same *ANN*. Thus, the evolutionary algorithm uses two different mutation operators (Fig. 2) with equal probabilities (50%): a *structural mutation* [12], which works by adding/deleting a random number (from 1 to M) of nodes or connections; and a *macro mutation*, which replaces a random number of weights (from 1 to M) by a new randomly generated value within the range $[-1.0, 1.0]$.



a) Original *MLP* b) After macro mutation c) After node deletion d) After adding connections

Fig. 2. Example of the application of the mutation operators

This algorithm will also be combined with a local optimization procedure, under a *Lamarckian* evolution setting [1]. In each generation, L epochs of the *RPROP* learning algorithm are applied to each individual (*MLP*) in the population, using the examples in the training set. In past work [10], this Lamarckian approach (with macro mutation) to training outperformed eight *evolutionary algorithms* (using different crossovers and mutations) and gradient-based algorithms (e.g. Backpropagation and RPROP).

The *fitness* function is based in the $RMSE$ (Eq. 1) computed over a validation set. The *selection* procedure is done by converting the fitness value into its ranking. Then, a roulette wheel scheme is applied, being used a substitution rate of 50%. Finally, the algorithm is stopped after G generations. The *SENN Ensemble (SENNE)* will be built using the best G individuals obtained during the evolutionary process, being the output computed as the average of the *MLPs*.

3 Results and Discussion

The *ANN/EC* experiments were conducted using a software package developed in *JAVA* by the authors. The other *DM/ML* techniques were computed using the *WEKA* software package with its default parameters [15]:

- *J48* – a classification decision tree based on the C4.5 algorithm;
- *M5P* – a regression decision tree (M5 algorithm);
- *IB5* – a *5-Nearest Neighbor*;
- *KStar* – an instance based algorithm; and
- *SVM* – a *Support Vector Machine*.

For each model, 10 runs of a 5-fold cross-validation process [4] (stratified in the classification tasks) were executed. This means that in each of these 50 experiments, 80% of the data is used for learning and 20% for testing.

With the pure *ANN* approaches, the learning data was divided into *training* (50% of the original dataset) and *validation* sets (30%). A different strategy was used for the *SENN*, since the simultaneous evolution of weights and topologies is very sensitive to overfitting. Thus, the validation set is divided into: a *fitness* set (15%), used for the fitness evaluation, and a *model selection* set (15%), used to select the best individual (or individuals when building an ensemble). The $SENN$ parameters were set to $P = 20$, $H = 10$, $L = 50$, $M = 5$ and $G = 20$. Tables 2 and 3 show the average errors of the 10 runs for each model and task. The last row of each table averages the global performance of each learning strategy.

When comparing the classification results, the *ANN* learning models (last four columns) are competitive, outperforming the other *ML* algorithms. In effect, the few exceptions are the **dermatology** and **sonar** tasks where the SVM and *KStar* get the best results. Regarding the neural approaches, the *SENN* excels the *HNN* with a 1.1% difference in the average performance. Moreover, the ensemble approaches (*HNNE* and *SENNE*) obtain better results when compared with the single based versions, with improvements of 1.4% and 1.2%. Overall, the *SENNE* obtains the best predictive accuracy, being the best choice in 5 tasks.

Table 2. The classification results ($PCCE$ values, in %)

Task	J48	IB5	KStar	SVM	HNN	SENN	HNNE	SENNE
Balance	78.1	87.6	88.3	87.7	94.8	96.1	95.7	**96.7**
Bupa	64.8	60.7	65.9	58.0	68.4	68.8	68.5	**69.0**
Car	91.3	92.3	87.1	93.5	97.4	98.5	98.3	**98.8**
Cmc	51.2	47.2	49.6	48.4	50.6	53.8	52.1	**54.5**
Dermatology	95.7	96.6	94.5	**97.4**	95.1	95.5	95.9	96.5
Ionosphere	89.4	84.6	84.0	87.9	88.9	89.9	**92.3**	91.9
Sonar	72.5	80.5	**85.2**	76.7	79.9	79.3	80.9	83.6
Yeast	56.0	57.1	53.1	56.6	58.2	59.9	59.9	**60.0**
Mean	74.9	75.8	76.0	75.8	79.1	80.2	80.5	**81.4**

Table 3. The regression results ($NRMSE$ values, in %)

Task	**M5P**	**IB5**	**KStar**	**SVM**	**HNN**	**SENN**	**HNNE**	**SENNE**
Abalone	24.3	25.3	24.8	25.0	23.2	23.2	23.2	**22.9**
Auto-mpg	**11.8**	15.1	14.6	13.5	14.2	12.8	12.2	12.2
Autos	**13.3**	21.4	21.6	13.8	14.6	15.2	14.2	13.6
Breast-cancer	40.8	40.8	43.6	44.0	42.6	42.4	47.4	**40.6**
Heart-disease	21.2	**21.0**	25.5	21.5	21.9	23.3	22.2	21.7
Housing	18.4	22.2	18.0	22.6	18.4	17.1	16.6	**15.8**
Servo	50.4	60.4	67.6	70.5	60.8	43.4	49.6	**38.3**
Wpbc	73.2	73.6	98.2	**71.7**	75.4	75.7	80.3	**71.7**
Mean	33.0	35.0	39.2	35.3	33.9	31.6	33.2	**29.6**

For the regression tasks, the decision tree (*M5P*) is quite competitive, outperforming all non evolutionary approaches. As before, the *SENN* excels the *HNN* (2.3% improvement) and the ensembles behave better, with enhancements of 0.7% and 2.0%. Thus, the best alternative is the evolutionary ensemble (*SENNE*), followed by its single based version (*SENN*).

Similar work has been reported in the literature, namely the EPNet system [6], which obtained interesting results. However, this approach was only applied to five UCI datasets where the best results are obtained by low complexity *MLPs* (in some cases linear models). It is not surprising that, since EPNet heavily promotes simple models, good results were obtained for these cases. In this work, the majority of the problems demanded *MLPs* with a much higher number of hidden nodes, where is it believed that the EPNet system would not excel.

4 Conclusions

In this work, a *Simultaneous Evolutionary Neural Network (SENN)* algorithm is proposed, aiming at the optimization of the neural structure and weights. This approach was enhanced by considering ensembles, which combine the best *ANNs* obtained by the *SENN* approach. The results obtained in several real-world classification and regression tasks confirm the competitive *SENN* performances, when compared with a heuristic trial-and-error design procedure (*HNN*) and with other *DM/ML* algorithms.

Another advantage presented by the *SENN* is the reduced computational effort, when compared to an evolutionary algorithm that performs only topology optimization [12]. Indeed, with the current setup, the computational burden is similar to the one required by the *HNN*. Overall, the hybrid EC/ANN ensemble *(SENNE)* presents the best predictive accuracy while requiring no extra computation, thus being the advised choice. In future work, it is intended to test similar techniques with other *ANNs* (e.g., *Recurrent Neural Networks*). Furthermore, more elaborated ensembles could be considered, by designing fitness functions which reward specialization [7].

References

1. P. Cortez, M. Rocha, and J. Neves. A Lamarckian Approach for Neural Network Training. *Neural Processing Letters*, 15(2):105–116, April 2002.
2. T. Dietterich. Ensemble methods in machine learning. In J. Kittler and F. Roli, editors, *Multiple Classifier Systems, LNCS 1857*, pages 1–15. Springer, 2001.
3. S. Haykin. *Neural Networks - A Compreensive Foundation.* Prentice-Hall, New Jersey, 2nd edition, 1999.
4. R. Kohavi. A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, Montreal, Quebec, Canada, August 1995.
5. T. Kwok and D. Yeung. Constructive algorithms for structure learning in feedforward neural networks for regression problems problems: A survey. *IEEE Transactions on Neural Networks*, 8(3):630–645, May 1999.
6. Y. Liu and X. Yao. Evolving Modular Neural Networks Which Generalize Well. In *Proc. of the 1997 IEEE Intern. Confer. on Evolutionary Computation, Indianapolis*, pages 670–675, New York, 1997. IEEE Press.
7. Y. Liu, X. Yao, and T. Higuchi. Evolutionary Ensembles with Negative Correlation Learning. *IEEE Transactions on Evolutionary Computation*, 4(4):380–387, 2000.
8. J.R. Quinlan. *Comparing Connectionist and Symbolic Learning Methods*, pages 445–456. MIT Press, Cambridge, Massachustess, 1994.
9. M. Riedmiller. Supervised Learning in Multilayer Perceptrons - from Backpropagation to Adaptive Learning Techniques. *Computer Standards and Interfaces*, 16, 1994.
10. M. Rocha, P. Cortez, and J. Neves. Evolutionary Neural Network Learning. In F. Pires and S. Abreu, editors, *Progress in Artificial Intelligence, EPIA 2003 Proceedings, LNAI 2902*, pages 24–28, Beja, Portugal, December 2003. Springer.
11. M. Rocha, P. Cortez, and J. Neves. Ensembles of Artificial Neural Networks with Heterogeneous Topologies. In *Proceedings of the 4th Symposium on Engineering of Intelligent Systems (EIS2004).* ICSC Academic Press, March 2004.
12. M. Rocha, P. Cortez, and J. Neves. Evolutionary Design of Neural Networks for Classification and Regression. In *ICANNGA Proceedings*, Coimbra, Portugal, March, 2005. Springer.
13. C. Soares. Is the UCI Repository Useful for Data Mining? In F. Pires and S. Abreu, editors, *Progress in Artificial Intelligence, EPIA 2003 Proceedings, LNAI 2902*, pages 209–223, Beja, Portugal, 2003. Springer.
14. G. Thimm and E. Fiesler. Evaluating pruning methods. In *Proc. of the Int. Symp. on Artificial Neural Networks*, pages 20–25, Taiwan, December 1995.
15. I. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations.* Morgan Kaufmann, USA, 2000.
16. X. Yao. Evolving Artificial Neural Networks. In *Proc. of the IEEE*, 87(9): 1423-1447, September 1999.
17. X. Yao and Y. Liu. A New Evolutionary System for Evolving Artificial Neural Networks. *IEEE Transactions on Neural Networks*, 8(3):694–713, 1997.