

# Evolving feed-forward neural networks through evolutionary mutation parameters

Annunziato M.<sup>1</sup>, Bertini I.<sup>1</sup>, Iannone R.<sup>2</sup>, Pizzuti S.<sup>1</sup>

<sup>1</sup>Energy New technologies and Environment Agency, 'Casaccia' R.C.

Via Anguillarese 301, 00060 Rome, Italy  
{mauro.annunziato, ilaria.bertini,  
stefano.pizzuti}@casaccia.enea.it

<sup>2</sup>University of Rome 'La Sapienza', Dept. of Computer Science

Via Salaria 113, Rome, Italy  
rob.iannone@gmail.com

**Abstract.** In this paper we show a preliminary work on evolutionary mutation parameters in order to understand whether it is possible or not to skip mutation parameters tuning. In particular, rather than considering mutation parameters as global environmental features, we regard them as endogenous features of the individuals by putting them directly in the genotype. In this way we let the optimal values emerge from the evolutionary process itself. As case study, we apply the proposed methodology to the training of feed-forward neural networks on nine classification benchmarks and compare it to other five well established techniques. Results show the effectiveness of the proposed approach to get very promising results passing over the boring task of off-line optimal parameters tuning.

## 1 Introduction

Artificial Neural Networks (ANN) and Evolutionary Algorithms (EA) are both abstractions of natural processes. They are formulated into a computational model so that the learning power of neural networks and adaptive capabilities of evolutionary processes can be harnessed in an artificial life environment. 'Adaptive learning', as it is called, produces results that demonstrate how complex and purposeful behavior can be induced in a system by randomly varying the topology and the rules governing the system. Evolutionary algorithms can help determine optimized neural network architectures giving rise to a new branch of ANN known as Evolutionary Neural Networks [30] (ENN). It has been found [1] that, in most cases, the combinations of evolutionary algorithms and neural nets perform equally well (in terms of accuracy) and were as accurate as hand-designed neural networks trained with backpropagation [9]. However, some combinations of EAs and ANNs performed much better for some data than the hand-designed networks or other EA/ANN combinations. This suggests that in applications where accuracy is a premium, it might pay off to experiment with EA and ANN combinations.

In this context methodological efforts ranged from the simple weights optimization to the simultaneous evolution of weights and topological neural structures [16][19][22][28]. ENN applications are wide as well and ranged from modelling (es. [3],[25]) to classification tasks (es. [10]).

However, all the proposed methodologies and applications consider the algorithm's parameters as static (at most adaptive) global environmental features and do not take into account the possibility of considering them as endogenous features of the individuals, although it is known that the evolutionary adaptation of mutation rates plays an important role in the molecular evolution of life [24].

In order to face this new challenge, evolutionary programming has started to study evolving mutation rates in an effort to automate control of evolutionary search for function optimisations. Preliminary efforts proved that automated control is feasible [5] [15], and continuing research is fine-tuning this process [12]. In particular, in [5] it is presented an approach in which a basic idea from Evolution Strategies (ESs) is transferred to EAs. Mutation rates, instead of being handled as a global constant external parameters, are changed into endogenous items which are adapting during the research process. In this work experimental results indicate that environment-dependent self-adaption of appropriate settings for the mutation rate is possible.

Later, it has been demonstrated [17] that energy dependent mutation rate adaptation can play a pivotal role in the evolution of complexity. In [7], through a model consisting of a two-dimensional world with a fluctuating population of evolving agents, it is provided evidence that evolving mutation rates adapt to values around a transition among qualitatively different kinds of complex adaptive systems (meta-stable, quasi-clonal systems, and randomly fluctuating systems). Results provide an especially simple illustration of how the evolution of evolvability creates and tunes the capacity of complex adaptive systems to generate order through adaptive evolution.

In this context, the goal of the proposed work is to study the effectiveness of using evolutionary mutation rates, applied to the evolution of weights in feed forward neural networks, in order to achieve good results without the off-line optimal parameter tuning process.

## **2 The evolutionary environment**

### **2.1 The Alife algorithm**

The implemented artificial environment is inspired by the 'Artificial Life (*ALIFE*)' methodology [20][21]. This approach has been tested for the optimisation of static well known benchmark problems, like the Travelling Salesman Problem, as well as real life problems [4]. The *ALIFE* context is a two-dimensional lattice (*life space*) representing a flat physical space where the artificial *individuals* can move around. At each iteration (or *life cycle*), individuals move in the life space and, in case of meeting with other individuals, interaction occurs. Each individual has a particular set of rules that determines its interactions with other agents basically based on a competition for

energy in relation to the performance value. Individuals can self-reproduce via haploid mutation and can occur only if the individual has an *energy* greater than a specific *birth energy*. In fact, during reproduction, an amount of energy equal to the birth energy is transferred from the parent to the child. In the haploid reproduction a probabilistic test for self reproduction is performed at every life cycle and a probabilistic-random mutation occurs on the genes according to the mutation rate and the mutation amplitude (see next paragraph). When two individuals meet, fighting occurs. The winner is the individual characterized by a greater value of performance and the loser transfers an amount of energy (*fighting energy*) to the winner. At every life cycle each individual *age* is increased and the age reaches a value close to the *average lifetime*, the probability of natural death increases. This ageing mechanism is very important to warrant the possibility to lose memory of very old solutions and follow the process evolution. Another mechanism of death occurs when the individual reaches the null energy due to reproduction or fighting with other individuals. In this way, when there is a reproduction event the population size increases of one unit while in a death event the population size decreases of one unit. Therefore, the population size is dynamic and it is limited by the dimension of the physical space. The reader interested in further details can refer to [2].

## 2.2 Implementation

In the genotype of the individuals it is stored the information concerning the problem to be solved and the individuals' fitness is calculated as function of the genotype itself. In the case we are facing here, each individual represents a feed forward neural network in competition with the others by means of the proper fitness, which depends on the capability of reconstructing the training database. The fitness of the individuals is measured referring to the global error in modelling the training database with the following formula:

$$\text{Fitness} = 1 - \text{RMSE} \quad (1)$$

Where *RMSE* is the classical Root Mean Squared Error (2) normalised in the lattice [0,1] used by the back-propagation (BP) algorithm. This cost function has been chosen in order to directly compare the results with the ones obtained with BP methodology.

$$\text{RMSE} = \sqrt{\frac{1}{n} \sum_1^n (0.5 * \sum_1^m (y - y_t)^2)} \quad (2)$$

Where *n* is the dimension of the training data set, *m* is the number of output neurons, *y* is the estimated output and *y<sub>t</sub>* is the corresponding target. All the inputs and outputs of the networks are normalized between 0 and 1 (as well as the targets) and there are no differences among the activation functions for hidden and output nodes. As measure of the level reached in the training stage, we take the fitness of the best individual corresponding to the best performing neural network.

The genotype is therefore composed (see table 1) by the network features (weights and biases) to be optimised, by the mutation rate (*Rate*), representing the number of

genes to be mutated during the reproduction stage, for weights, biases and, associated to each gene, by the highest mutation amplitude (*Delta*). Therefore, during evolution both topological features (solutions) and parameters are simultaneously evolved.

**Table 1.** Individuals' genotype

GENOTYPE	
Solutions	Parameters
Weight 1	Rate
...	Delta(1)
...	...
...	...
Weight $k$	Delta( $k$ )
Bias 1	Delta( $k+1$ )
...	...
...	...
Bias $h$	Delta( $k+h$ )

### 3 Experimentation

Experimentation concerned the optimal training of feed forward neural networks in order to solve nine classification benchmarks taken from the UCI repository [8].

The neural optimisation task has been accomplished using the proposed evolutionary approach and compared to the following five well established methodologies: Multilayer Perceptron (*MLP*) [26] trained with the Back-Propagation algorithm, Kstar[11], MultiBoost (*MB*)[27], Voting Feature Interval (*VFI*) [14] and Particle Swarm Optimisation (*PSO*) [18]. For the first four the WEKA tool [29] has been used and for PSO we took the results presented in [13]. Each data set (see table 2) has been split in two parts: training (75% of the whole data set) and testing set (25% of the whole data set).

**Table 2.** Data sets features

Problem	Data set size	Training set Size	Testing set Size	Classes	Input size
Card	690	517	173	2	51
Diabetes	768	576	192	2	8
Glass	214	160	54	6	9
Heart	303	227	76	2	35
Horse	364	273	91	3	58
Iris	150	112	38	3	4
Wdbc	569	426	143	2	30
WdbcInt	699	524	175	2	9
Wine	178	133	45	3	13

In our experimentation we compared the mentioned techniques to the evolutionary neural networks trained with (*EvoNN*) and without (*ENN*) the evolutionary mutation parameters. For these two experimentations we used a physical space of 25X25 cells corresponding to a maximum population size of 625 individuals. The neural topologies used in the *EvoNN*, *ENN* and *MLP* tests are reported in table 3 and results are in table 4 where we show the average of the classification error (percentage) on the testing set. For each of the mentioned techniques, we performed ten runs on each problem. Moreover, to avoid over-fitting in the MLP and ENN tests, we used the early stopping criterion trying different number of generations and then choosing the number of iterations which gave the best testing results. The same number of generations were therefore used in the *EvoNN* test. On average, the best ENN and *EvoNN* experimentations needed about 300000 performance evaluations.

All the other parameters of the other techniques were not optimised because the goal of this work is to get results without the off-line tuning of parameters which, in this case, are optimised during the evolutionary process (figure 1).

**Table 3.** Neural topologies

Problem	Neural Topology (input-hidden-output)
Card	51-4-2
Diabetes	8-3-2
Glass	9-2-6
Heart	35-4-2
Horse	58-4-3
Iris	4-4-3
Wdbc	30-4-2
WdbcIn	9-3-2
Wine	13-3-3

**Table 4.** Experimental testing results (classification error)

	EvoNN	MLP	KSTAR	ENN	PSO	MB	VFI
Card	17.98%	16.7%	24.28%	24.1%	22.84%	<b>13.8%</b>	25.43%
Diabete	<b>21.82%</b>	21.9%	32.29%	22.1%	22.5 %	26.5%	54.69%
Glass	35.00%	37%	<b>25.93%</b>	40.3%	41.69%	61.1%	46.30%
Heart	14.87%	17.1%	25.00%	16.4%	17.46%	<b>10.5%</b>	18.42%
Horse	38.24%	38.4%	<b>34.07%</b>	36.1%	40.98%	36.2%	42.86%
Iris	<b>2.63%</b>	<b>2.63%</b>	5.26%	7.11%	<b>2.63%</b>	7.90%	7.90%
Wdbc	<b>1.75%</b>	2.10%	5.59%	3.15%	5.73%	2.80%	5.59%
WdbcIn	<b>1.09%</b>	1.71%	1.14%	3.54%	2.87%	4.00%	1.71%
Wine	3.33%	<b>2.22%</b>	<b>2.22%</b>	4.22%	4.44%	22.2%	11.11%
Average	<b>15.19%</b>	15.5%	17.31%	17.4%	17.91%	20.5%	23.78%

These results show the effectiveness of the proposed methodology based on evolutionary mutation parameters. In fact, this method provides the best global

performance obtaining the best results on four problems. In particular, it is interesting the comparison with the MLP ANNs, trained with the Back-Propagation Algorithm, and the original ENN with constant mutation parameters. This comparison directly shows the performance improvement when using the suggested technique.

As regards cpu time, KSTAR, MB and VFI are very fast (1-2 seconds) because they are statistical clustering techniques which do not require a training stage. For PSO we got the results from [13] which does not report such an information and for the other methods the cpu training time ranges from 25 to 190 seconds.

After all, in figure 1 we can see how the mutation parameters evolve compared to the performance behaviour. From this graph it is interesting to point out that after several generations the optimal parameter values emerge and tend to stabilize as well as the performance. This is remarkable because it is the evidence that the evolutionary environment is able to optimise the problem itself and to find the best parameters for that problem avoiding the off-line tuning of the parameters.



**Fig. 1.** Evolution of mutation parameters

## 4 Conclusion

In this paper we showed a preliminary work on how evolutionary mutation parameters affect the performance of the optimization in an evolutionary environment. In particular, rather than considering the mutation parameters as global environmental features, we regarded them as endogenous features of the individuals by putting them directly in the genotype.

The final goal of this work is that we can achieve very good results without the need of parameters tuning because, making them evolutionary, their optimal values emerge from the evolutionary optimization itself.

As case study, we applied the proposed methodology to the training of feed-forward neural networks. We tested it on nine classification benchmarks and compared it to other five well established techniques. Results on testing data showed that the proposed methodology performs globally better than the others achieving the best results on four problems.

In the end, we studied the evolution of the mutation parameters and we found out they tend to emerge and stabilize around some optimal values as well as the achieved performance. These preliminary results are extremely encouraging and future work will focus on finding out a formal justification of the solution, on testing the proposed approach on real problems and on comparing the evolutionary environment to other similar techniques (like genetic algorithms).

## References

1. Alander, J. T.: An indexed bibliography of genetic algorithms and neural networks. Technical Report 94-1-NN, University of Vaasa, Department of Information Technology and Production Economics (1998).
2. Annunziato, M., Bertini, I., Lucchetti, M., Pannicelli, A. and Pizzuti, S. : Adaptivity of Artificial Life Environment for On-Line Optimization of Evolving Dynamical Systems, in Proc. EUNITE01, Tenerife (Spain) (2001).
3. Annunziato, M., Bertini, I., Pannicelli, A., Pizzuti, S. : Evolutionary feed-forward neural networks for traffic prediction , proceedings of EUROGEN2003, Barcelona, Spain (2003)
4. Annunziato M. , Lucchetti M., Orsini G., Pizzuti S. : ARTIFICIAL LIFE AND ONLINE FLOWS OPTIMISATION IN ENERGY NETWORKS, IEEE Swarm Intelligence Symposium, Pasadena (CA), USA, 2005
5. Bäch, T. : Self-adaptation in genetic algorithms", in *Towards a Practice of Autonomous Systems*, F. J. Varela & P. Bourgine (Eds.), Cambridge, MA: Bradford/MIT Press, (1992) 263-271.
6. Balakrishnan, K. and Honavar, V : Evolutionary Design of Neural Architectures - A Preliminary Taxonomy and Guide to Literature, Technical Report CS TR95-01, Dept. of Computer Science, Iowa State University (1995)
7. Bedau, M. A. and Seymour R.: Adaptation of Mutation Rates in a Simple Model of Evolution, *Complexity International*, vol.2, (1995)
8. Blake, C. L and Merz, C. J. : UCI repository of machine learning databases, University of California, Irvine, <http://www.ics.uci.edu/~mllearn/MLRepository.html> (1998)
9. Cant-Paz, E. and Kamath, C. : An empirical comparison of combinations of evolutionary algorithms and neural networks for classification problems. *IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics* (2005) 915-927.
10. Cantú-Paz, E. and Kamath, C. : Evolving neural networks for the classification of galaxies," *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2002*, Morgan Kaufmann Publishers, San Francisco (CA), (2002) 1019-1026
11. Cleary, J. G. and Trigg, L. E. : K\*: An Instance- based Learner Using an Entropic Distance Measure, *Proceedings of the 12th International Conference on Machine learning* (1995) 108-114

12. Davis, M. W.: The natural formation of gaussian mutation strategies in evolutionary programming, Proceedings of the 3rd Annual Conference on Evolutionary Programming, A. V. Sebald & L. J. Fogel (Eds.), River Edge, NJ: World Scientific (1994).
13. De Falco, I. , Della Coppa, A. and Tarantino, E. : Impiego della particle swarm optimization per la classificazione in database, II Italian Artificial Life Workshop, Rome, Italy, ISTC-CNR (2005)
14. Demiroz, G. and Guvenir, A. : Classification by voting feature intervals, ECML-97 (1997)
15. Fogel, D. B., Fogel, L. J. & Atmar J. W.: Meta-evolutionary programming, Proceedings of the 25th Asilomar Conference on Signals, Systems and Computers, R. R. Chen (Ed.), San Jose: Maple Press, (1991) 540-545.
16. Hwang, M. W., Choi, J. Y., and Park, J.: Evolutionary projection neural networks, in Proceedings of the 1997 IEEE International Conference on Evolutionary Computation, ICEC'97, (Piscataway, NJ, USA), IEEE Press, (1997) 667-671.
17. Jan, T. and Kim : Energy Dependent Adaptation of Mutation Rates in Computer Models of Evolution, Proceedings of ALIFE VI, Los Angeles, CA, (1998)
18. Kennedy, J. and Eberhart R.C. : Particle swarm optimization. Proc. IEEE International Conference on Neural Networks, IV. Piscataway, NJ: IEEE Service Center, (1995) 1942–1948
19. Kenneth, O. Stanley and Miikkulainen, R. : Evolving Neural Networks Through Augmenting Topologies (2002)
20. Langton, C. G. : Artificial life in Artificial life, C. Langton (Ed.). Addison-Wesley. (1989)
21. Langton, C. G. : The Garden in the Machine, The Emerging Science of Artificial Life, Princeton University Press (1989)
22. Liu, Y. and Yao, X., : Evolutionary design of artificial neural networks with different nodes, in Proc. of the 1996 IEEE Int'l Conf. on Evolutionary Computation (ICEC'96), Nagoya, Japan, IEEE Press, New York, NY 10017-2394, (1996) 670-675
23. Matteucci, M. : ELeaRNT: Evolutionary Learning of Rich Neural Network Topologies, Technical Report N CMU-CALD-02-103, Department of Computer Science - Carnegie Mellon University, Pittsburgh PA (2002)
24. Metzgar, D. and Wills, C. : Evidence for the Adaptive Evolution of Mutation Rates, Cell, Vol. 101, (2000) 581-584
25. Prudencio, R.B.C. and Ludermir, T.B. : Evolutionary Design Of Neural Networks: Application To River Flow Prediction, Proceedings of the IASTED International Conference on Artificial Intelligence and Applications, AIA 2001, Marbella, Spain (2001)
26. Rumelhart, D. E., Hinton, G. E., and Williams, R. J. : Learning representations by backpropagating errors. Nature, 323 (1986) 533–536
27. Webb, G. I. : MultiBoosting: a technique for combining boosting and wagging, Machine Learning, vol. 40 (2), (2000) 159-196
28. White, D. and Ligomenides, P.: GANNet: a genetic algorithm for optimizing topology and weights in neural network design, in Proc. of Int'l Workshop on Artificial Neural Networks (IWANN'93), Springer-Verlag, Lecture Notes in Computer Science, Vol. 686 (1993) 322-327
29. Witten, I. H. and Frank, E.: Data mining: practical machine learning tool and technique with Java implementation, San Francisco: Morgan Kaufmann (2000)
30. Yao, X., “Evolving Artificial Neural Networks”, Proceedings of the IEEE, 87(9): (1999)1423-1447

## Reviews

1. It is a very preliminary work.
2. Authors include 4 self references !
3. No CPU time comparison is made of the proposed methods.
4. Abstract is too short.
5. Figure 1 is irrelevant.
6. Some words ("wrong classification error") are confusing.
  
7. Interesting paper. A more formal justification of the solution could convince better the reader on the value of the method.

## Updates

1. it has been remarked that it actually is a preliminary work
2. 3 self references have been removed (6 to 3)
3. info about cpu time has been provided
4. the abstract has been expanded
5. it has been stressed the importance of figure 1
6. confusing words have been changed
7. formal justification as been included as future work