# Lamarckian Training of Feedforward Neural Networks *

Paulo Cortez Miguel Rocha and José Neves

pcortez@di.uminho.pt mrocha@di.uminho.pt jneves@di.uminho.pt
Departamento de Informática, Campus de Gualtar,
Universidade do Minho, 4710-057 Braga, PORTUGAL

**Abstract**. Living creatures improve their adaptation capabilities to a changing world by means of two orthogonal processes: *evolution* and *lifetime learning*. Within *Artificial Intelligence*, both mechanisms inspired the development of non-orthodox problem solving tools, namely *Genetic and Evolutionary Algorithms (GEAs)* and *Artificial Neural Networks (ANNs)*. Several local search gradient-based methods have been developed for *ANN* training, with considerable success; however, in some situations, such procedures may lead to local minima. Under this scenario, the combination of *evolution* and *learning* techniques, may lead to better results (e.g., global optima). Comparative tests on several *Machine Learning* tasks attest this claim.

**Keywords:** Genetic and Evolutionary Algorithms, Feedforward Neural Networks, Lamarckian Optimization.

## 1 Introduction

The remarkable adaptation of some living creatures to their environments comes as a result of the interaction of two processes, working at different time scales: *evolution* and *lifetime learning*. *Evolution* is a slow stochastic process that takes place at the population level and determines the basic structures of an organism. *Lifetime learning* works by tuning up the structures of an individual, by a process of gradual improvement of the individual's adaptation to its surrounding environment. In terms of a computational procedure, *evolution* seems suitable for global search, while *learning* should be used to perform local search. New optimization procedures, based on analogies with the evolution of natural living systems, have been carried out, giving rise to techniques such

*The work of Paulo Cortez was supported by *FCT* through the grant PRAXIS XXI/BD/13793/97. The work of José Neves was supported by the PRAXIS' project PRAXIS/P/EEI/13096/98.

as the *Artificial Neural Networks (ANNs)* and *Genetic and Evolutionary Algorithms (GEAs)*, which have already on their shoulders interesting results on a broad set of scientific and engineering problems.

The *Feedforward Neural Network (FNN)* is one of the most popular *ANN* architectures, where neurons are grouped in layers and only forward connections exist. This provides a powerful connectionist model that can learn any kind of continuous nonlinear mapping, with successful applications such as *Time Series Forecasting*, *Medical Diagnostics* or *Handwritten Recognition*, just to name a few. The interest in supervised learning and *FNNs* was stimulated by the advent of *Backpropagation* algorithm [6]; since then several variants have been proposed, such as the *Quickprop* and the *RPROP* [3]. However, these may not escape from local minima when the error surface is rugged.

On the other hand, *GEAs* are suited for combinatorial problems, where the exhaustion of all possible solutions requires huge computation. *GEAs* perform a global multi-point search, being able to escape from undesired local minima. The use of evolutionary search may overcome gradient-based handicaps, but convergence is in general much slower, since these are general purpose methods.

*Evolution* and *learning* can be combined in two major ways, namely the *Baldwin* and *Lamarckian Evolution*. Both approaches use *lifetime learning* to accelerate *evolution*. The main difference is that the latter allows the inheritance of the acquired information. The aim is to study the benefits of the combination of *evolution* and *lifetime learning*, when applied to *Machine Learning* tasks. The *Lamarckian* point of view will be adopted, since previous work favored this strategy under static environments [5]. The combination of *GEAs* and *ANNs* will be materialized via the evolution of a population, where each individual codes for the weights of an *FNN*. The individuals are allowed to improve their fitness during lifetime, by a gradient descent process.

## 2 Learning Models

Four different models will be defined to approach each learning task, namely:

**Connectionist Model (CM)** - The learning is achieved by a single *FNN*, with a fixed topology, with one hidden layer, using the logistic activation function. The initial weights are randomly assigned within the range $[-1; 1]$. The training is achieved by the *RPROP* algorithm, chosen due to its faster convergence and stability in terms of parameter's adjustment.

**Darwinian Model (DM)** - The learning process is accomplished by a *GEA*, where a population of 20 real-valued chromosomes is evolving, each coding for the weights of an *FNN*. In each iteration, 50% of the individuals are kept from the previous generation, being the remaining bred through the application of genetic operators. In this work, two operators were adopted, namely the *two point crossover* [1] and a *gaussian mutation* that adds, to a given gene, a value taken from a gaussian distribution, with zero mean. The fitness of each chromosome is calculated by an error

metric, the *Root Mean Squared Error (RMSE)*, that ranges over all the training patterns.

**Lamarckian Model (LM)** - it combines both *lifetime learning* and *evolution*, making use of *GEAs*, as the main engine, and gradient-based *FNN* training methods, for local search (in this case, 20 epochs of the *RPROP* algorithm). The improved weights are encoded back into the chromosome (Figure 1).

**Population of Connectionist Models (PM)** - This model is approach is added with the purpose of achieving a fair comparison among models, in particular to measure the weight of the genetic operators. A population of 20 *ANN's* is evolved with the application of the *RPROP* algorithm.

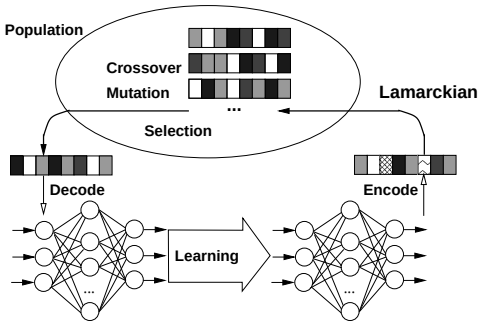


Figure 1: An illustration of the Lamarckian strategy of inheritance

# 3 Machine Learning Tasks

In the experiments carried out in this work, two artificially created tasks and two real ones were selected:

**N Bit Parity (NBP)** - This is a famous benchmark [3], being defined by $2^N$ patterns of $N$ inputs and one output, which is set to the value 1, if the total number of input bits set to 1 is odd, and 0 otherwise.

**Three Color Cube (TCC)** - This is a simple artificial *ML* task that consists in learning how to paint a large 3D cube, made up by a 3x3 grid of blocks (27 smaller cubes) (Figure 2) [5]. Each smaller cube is represented by its coordinates on the $X$, $Y$ and $Z$ axis, that can take values from $\{-1, 0, 1\}$, and can be painted with three different colors: black, grey and white. The corners are black, the cubes in the center are white, being the others grey (Figure 2). In terms of the *ANN* training cases, 27 patterns are created, one for each cube, consisting of 3 inputs and 3 outputs (one for each color).

**Sonar: Mines vs Rocks (SMR)** - The task is to discriminate between sonar signals bounced off a metal cylinder and those bounced of a roughly cylindrical rock [2]. The data has 104 training cases with 60 real inputs and one boolean output.

**Diabetes in Pima Indians (DPI)** - This task consists in diabetes diagnosis (a boolean output) from seven input real variables (e.g., number of pregnancies). The data is defined by 200 samples, taken from a population of women of a Pima Indian heritage [4].

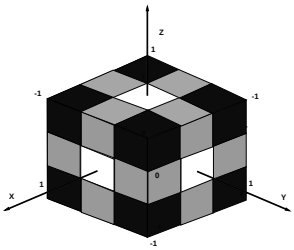


Figure 2: The *Color Cube Problem*

# 4 Experiments and Results

All experiments reported in this work were conducted using programming environments developed in *C++*, under the *Linux* operating system. The results obtained are compared in terms of two orthogonal parameters, the overall learning's *accuracy*, measured by the *RMSE* obtained for the set of patterns, and the process *efficiency*, measured by the time elapsed (in seconds). For all models, at each time slot, it were considered the average of the results obtained in thirty independent runs. For the last three learning models, the best individual in the population was considered. The number of hidden nodes was set to seven for the *DPI* task, and six for the other benchmarcks (*6BP*, *TCC* and *SMR*).

The results obtained are shown in Figure 3, where the evolution of the *RMSE* is plotted, in terms of *CPU* time (in seconds). More accurate information on the final results is presented in Table 1. An analysis of the results shows that the *LM* behaves in a better way, although the *CM* presents a faster convergence in the initial stages, but looks as being trapped in local minima. The results obtained by the *PM* show the importance of the genetic recombination. In fact, it is not enough to introduce diversity to escape from local minima. The performance of the *PM* was even worst than that of the *CM* one due to the sharing of computational resources between the several *ANNs*. Therefore, the combination of *lifetime learning* and *evolution* may exceed the sum of its parts.

Table 1: Best learning results.

| Task | CM | DM | LM | PM |
|---|---|---|---|---|
| *6BP* | 0.236 | 0.315 | **0.109** | 0.247 |
| *TCC* | 0.280 | 0.339 | **0.226** | 0.248 |
| *SMR* | 0.051 | 0.378 | **0.014** | 0.314 |
| *DPI* | 0.171 | 0.340 | **0.142** | 0.192 |

# 5 Conclusions and Future Work

Some work in this arena has already been put forward, where similar models have been compared. However, most of these studies consider only the benefits of *lifetime learning*, and the tradeoff between benefits and costs is rarely considered. In the present work, the comparisons between the models is made by considering the CPU time, so that they can be fair.

The results do support the idea that the *Lamarckian* evolution of learning entities makes itself has a very interesting method for *Machine Learning*. In the future one intends to enlarge the experiments domain, by looking at some real-world applications, such as those of *system's control*, *time-series forecasting* or *production scheduling*.

# References


[1] D. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Publishing Company, Inc., NY, USA, 1989.

[2] R. P. Gorman and T. J. Sejnowski. Analysis of Hidden Units in a Layered Network Trained to Classify Sonar Targets. *Neural Networks*, 1:75–89, 1986.

[3] M. Riedmiller. Supervised Learning in Multilayer Perceptrons - from Backpropagation to Adaptive Learning Techniques . *Computer Standards and Interfaces*, 16, 1994.

[4] B. D. Ripley. *Pattern Recognition and Neural Networks*. Cambridge University Press, 1996.

[5] M. Rocha, P. Cortez, and J. Neves. The Relationship between Learning and Evolution in Static and Dynamic Environments. In *Proceedings of the Second International Symposium on Engineering of Intelligent Systems (EIS2000)*, Paisley, Scotland, jul 2000.

[6] D. Rumelhart, G. Hinton, and R. Williams. Learning Internal Representations by Error Propagation. In D. Rulmelhart and J. McClelland, editors, *Parallel Distributed Processing: Explorations in the Microstructures of Cognition*, volume 1, pages 318–362, MIT Press, Cambridge MA, 1986.

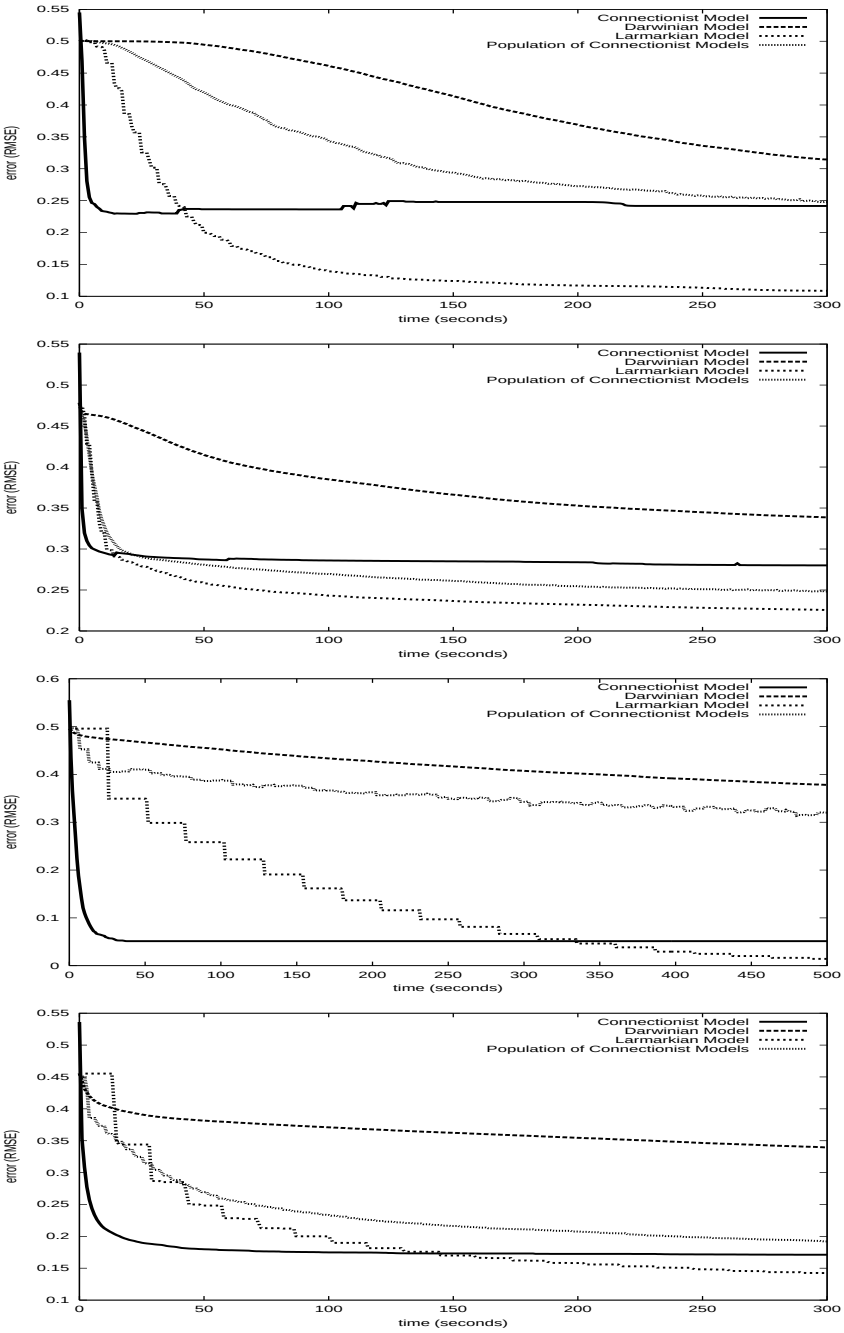


Figure 3: Results for the *6BP*, *TCC*, *SMR* and *DPI* tasks.