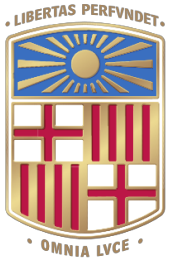
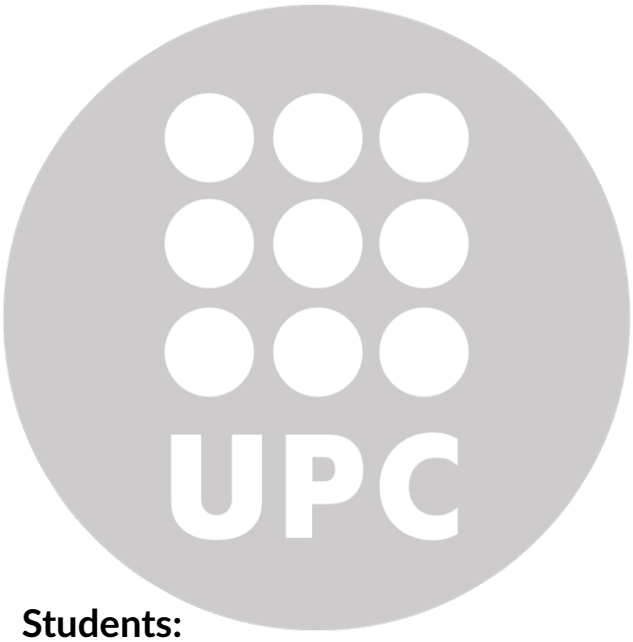


Laboratory Exercise 2: Training Neural Networks with Evolutionary Algorithms

Computational Intelligence

Master in Artificial Intelligence



Students:
- Alam López
- Mario Rosas

Professor: Lluís Belanche

Academic Year: 2023-1

Laboratory Group: 12

Dec 28, 2023

Contents

Training Neural Networks with Evolutionary Algorithms

1 | Abstract

In the current work, we present a comparison of three different Artificial Neural Network (ANN) weights optimization methods; two are based on Evolutionary Algorithms, and the third and baseline the well-known derivative approach of backpropagation. We have tested these methods on three synthetic datasets of binary classification problems with an increasing level of difficulty. Results demonstrated that the tuning of neural networks weights improves significantly their performance in terms of error. In the post-hoc analysis there were found significant differences between the computational time.

2 | Introduction

Evolutionary Algorithms (EAs), are known for their unique approach that mimics species in nature and their survival to solve complex computational problems. Also, in an Artificial Intelligence hot topic, like Deep Learning, they bring some advantages in optimizing Artificial Neural Networks (ANNs), when compared with the traditional derivative methods. They do not need to meet some specific requirements like the objective function to be continuous, differentiable, or not noisy.

However, ¿Are they suitable in all the scenarios for this task? ¿Can we justify in terms of error and/or computational time the use of such an approach compared with the traditional one? The purpose of this work is to solve these questions by comparing an Evolution Strategy, a Genetic Algorithm, and a Backpropagation approach, in a small ANN with a simple binary classification task with small datasets of 100 instances each.

With the previous theoretical knowledge of EAs we expect that, in a task that is low in complexity but meaningful and comparable to simple real-world cases, all optimization methods will reach similar results in terms of error, even that are not statistically different when compared with backpropagation. However, we do expect them to have a statistically worse performance on the computational time, given the computational requirements of the EAs.

3 | Experimental setup

Since there are multiple different approaches on how to perform the evolutionary algorithms in artificial neural networks, and being an introductory course in the subject, we decided to focus our work on identifying the main differences between the three approaches, when optimizing the weights of an Artificial Neural Network.

By saying this, we mean that the architecture of the network (layers, neurons per layer, etc.), the hyperparameter, activation functions, etc. are fixed. The reason of the previous decision was to keep the work time manageable, and to understand the basics when having just one variable, which in this case is the matrix of weights. From this we determined that given the nature of the data a small architecture for the neural network would be sufficient to solve the task. Therefore our architecture is composed as shown in the figure 2.1, where we have the input layer, two hidden layers with 4 neurons each, and since they are all binary classification problems, the output layer has only one neuron. The activation function for all neurons in the hidden layers is the well-known ReLU, which is recommended for this type of layer, as it alleviates the saturation problem presented by other activation functions. The output layer neuron uses the sigmoid activation function, a very common choice when dealing with binary classification problems.

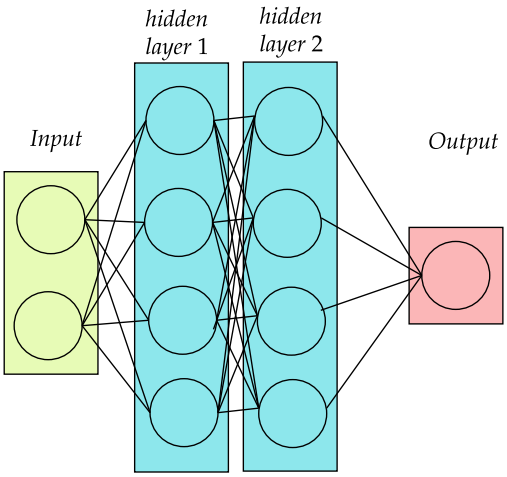


Figure 2.1: ANN architecture to be optimized.

We considered all the suggestions to keep the process as simple as possible to get isolated insights:

- We created 3 synthetic datasets, of 100 instances, with a binary classification problem with 2 features and 2 classes each, with an increasing level of difficulty
- We used a single ANN: a MLP with a field architecture 2,4,4,1.

- A single validation set was used to finetune hiperparameters.

As seen in class, there are a lot of parameters that can be configured in order to get the best performance of a model, this brings the possibility to explore and adjust the model according to the main metrics we want to achieve. For this reason, it is important to understand, how these parameters can affect the results, for us to be able to design a tailor-made model for a particular problem. For the execution of this task, we decided to develop the implementation in Python and use some libraries that already have the main algorithms implemented.

3.1 | Neural network implementation

The design and implementation of the neural network for this binary classification task involved the use of the Keras library. Keras was chosen for its flexibility and user-friendly features, which greatly expedited the process of creating a multilayer perceptron tailored to the task. The use of Keras not only facilitated the initial neural network design, but was also instrumental in the subsequent optimization steps. In particular, the manual extraction and updating of the weights, a crucial component in order to carry out the optimization process using the evolutionary strategy as the genetic algorithm. This choice of library allowed a cohesive integration of neural network design and evolutionary optimization, providing a good basis for exploring the performance of the model in the context of the task to be solved.

3.2 | Optimization method implementation

3.2.1 | Evolutionary strategy implementation

For the optimization by means of the evolutionary strategy we used the implementation made in the Evostra library, which was slightly modified to obtain the best weights through all the iterations, and to be able to obtain the history of values of the fitness function and to be able to obtain the corresponding graph. For this case certain parameters were set for all experiments including: learning rate, population size and the sigma value.

The Evolution Strategy (ES) implemented in Evostra is a stochastic optimization algorithm designed The algorithm is initialized with a set of weights and systematically explores a population of disturbed solutions, evaluating their performance using a reward function. The iterative process consists of generating several candidate solutions, normalizing the rewards and updating the parameters, in this case of the neural network, based on a weighted sum of successful perturbations. The algorithm allows parallel processing to improve efficiency and also allows dynamic adjustment

of the learning rate throughout the iterations, which contributes to the adaptability and effectiveness of the optimization process. In this case the fitness function is the prediction made by the forward step of the neural network using the weights updated by the evolutionary strategy.

3.2.2 | Genetic algorithm implementation

Fot the Genetic Algorithm implementation we used theh library which is PyGAD a versatile genetic algorithm library designed to allow to customize optimization solutions for a variety of problems. The library accommodates diverse problem spaces by allowing custom fitness functions. In the context of training neural networks, PyGAD requires also a user-defined fitness function that leverages the prediction computed from where the classification accuracy is extracted, serving as a measure of the solution's effectiveness. After each generation, the fitness function is invoked to evaluate solutions, and the genetic algorithm evolves the population by updating weights. A key feature is the callback function that facilitates the update of the trained weights attribute of network layers for each solution in the population. This process ensures that the evolved weights contribute to the ongoing optimization process, and the library includes safeguards to prevent premature convergence by maintaining diversity in the population.

The implementation further addresses parent selection, a crucial step in genetic algorithms, to drive the convergence rate. Various strategies such as tournament selection are employed to ensure the selection of diverse parents, preventing dominance by a single solution. PyGAD offers both generational and steady-state genetic algorithm variants. In steady-state mode, the one whish is used, only a few individuals are replaced at a time, maintaining the majority of the population across generations.

3.2.3 | Derivative based implementation

In the derivative-based optimization mode the function trains the neural network using the backpropagation algorithm. The function then sets the input data and labels from the training data set provided and compiles the model using Stochastic Gradient Descent (SGD) as the optimizer with the specified learning rate, impulse and nesterov parameters. The training process proceeds through the Keras fitting method, iterating over the data set for the specified number of epochs. During training, the function monitors and records accuracy and loss metrics for each epoch, creating visualizations representing learning trends.

Stochastic Gradient Descent (SGD) coupled with Backpropagation is a widely used optimization method in neural network training. SGD is an iterative optimization algorithm that updates the model parameters by calculating the gradient of the loss function with respect to the parameters and

adjusting them in the opposite direction of the gradient. This process is repeated for each training example or for a group of examples, providing a computationally efficient way to minimize the loss function. Backpropagation is the underlying algorithm for efficiently computing gradients across network layers. The combination of SGD and Backpropagation offers several advantages, such as scalability to large datasets, computational efficiency, and suitability for online learning scenarios. By iteratively updating model parameters based on training examples, SGD with Backpropagation enables neural networks to learn from diverse data sources and adapt to complex patterns, making it a foundational optimization method in the field of deep learning.

4 | Experimental Results

In setting up the final experiments, a critical phase was the systematic determination of fixed hyperparameters to underpin reliable and reproducible results over multiple experimental iterations. Several experiments were conducted to discern the optimal hyperparameter settings that would yield consistent results. Each experiment subsequently incorporated fixed parameters tailored to the specificities of each optimization method and data set. To ensure robustness, once the parameters were fixed, these experiments were meticulously run ten times each combination of dataset and optimization method. This deliberate repetition, combined with different random initializations of the parameters in each run, was intended to mitigate the impact of random variability on the results. The aggregated results, obtained by averaging the results of the ten runs, not only increase the reliability of our findings, but also provide an overall understanding of the consistent performance of each optimization method.

Tables reflecting the results obtained as well as some of the graphs generated are presented below, commenting on the findings of the results provided by each of the optimization algorithms and their effectiveness across various data sets and configurations. This approach facilitates a clear interpretation of the relative strengths and weaknesses inherent in each optimization method.

More data on the results can be found in the **results folder** of our implementation.

Table 2.1: Results for the Circles Dataset

Method	Train Acc	Train Error	Val Acc	Val Error	Test Acc	Test Error	Train Time	Inference Time
BP	0.5	7.712	0.533	7.198	0.467	8.227	55.370	0.119
ES	0.929	7.689	0.933	7.721	0.8	7.727	487.271	0.087
GA	0.929	7.689	0.933	7.721	0.733	7.655	438.694	0.098

Table 2.2: Results for the Classes Dataset

Method	Train Acc	Train Error	Val Acc	Val Error	Test Acc	Test Error	Train Time	Inference Time
BP	0.957	7.689	0.933	7.727	0.867	7.525	3.895	0.123
ES	0.957	7.692	0.933	7.727	0.8	7.459	220.129	0.060
GA	0.986	7.690	0.933	7.727	0.867	7.655	120.307	0.082

Table 2.3: Results for the Moons Dataset

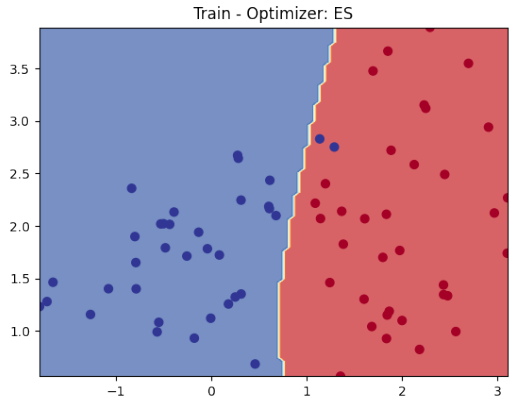
Method	Train Acc	Train Error	Val Acc	Val Error	Test Acc	Test Error	Train Time	Inference Time
BP	0.943	7.690	0.867	7.798	0.933	7.590	52.917	0.104
ES	0.914	7.686	0.933	7.584	0.867	7.786	358.968	0.083
GA	0.914	7.692	0.8	7.870	0.867	7.525	257.233	0.084

The results for the Moons dataset reveal interesting trends across optimization methods. Backpropagation (BP) attains the highest test accuracy of 93.3%, showcasing its effectiveness in capturing the underlying patterns of the complex dataset. Evolution Strategy (ES) and Genetic Algorithm (GA) exhibit slightly lower test accuracies of 86.7%, indicating a moderate level of success in generalization. While ES demonstrates efficiency in terms of training time, GA strikes a balance between accuracy and computational cost. These findings underscore the nuanced trade-offs among optimization methods, emphasizing the importance of considering both performance metrics and computational efficiency when selecting an algorithm for specific datasets.

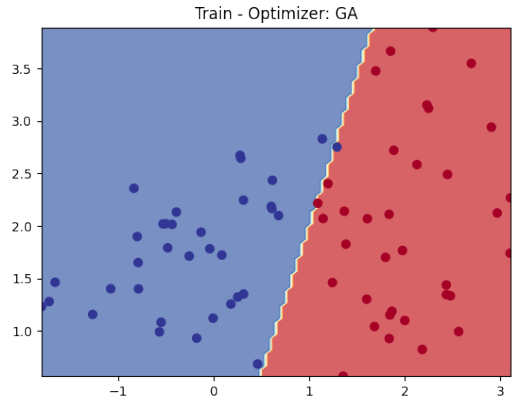
4.1 | Decision boundary plots.

When examining the results of the decision boundaries generated by the optimisation methods, an intriguing observation emerges: the decision boundaries produced by Evolution Strategies (ES) showed a generally less smooth profile compared to those generated by Genetic Algorithms (GA). This disparity in smoothness suggests that ESs, in navigating the optimisation landscape, may have encountered difficulties in achieving a smooth transition between different regions of the feature space. The relatively irregular nature of ES-derived decision boundaries could mean greater sensitivity to local variations in the objective function, which could lead to suboptimal generalisation in certain contexts.

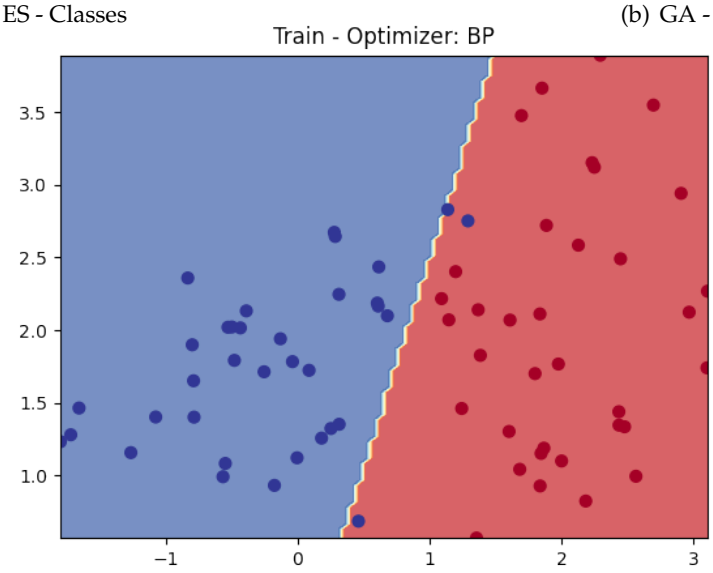
In contrast, the decision boundaries created by Genetic Algorithms seemed to manifest a smoother and more continuous character. This smoother profile implies that GAs, taking advantage of their evolutionary principles, managed to explore and exploit the solution space in a way that resulted in more coherent decision boundaries. The greater smoothness observed in GA decision boundaries could indicate an ability to better capture underlying patterns in the data, potentially contributing to



(a) ES - Classes



(b) GA - Classes

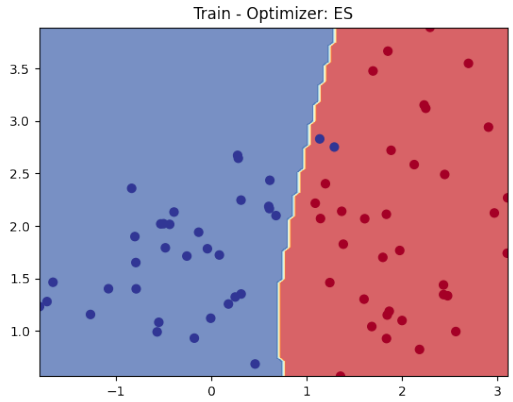


(c) SGD - Classes

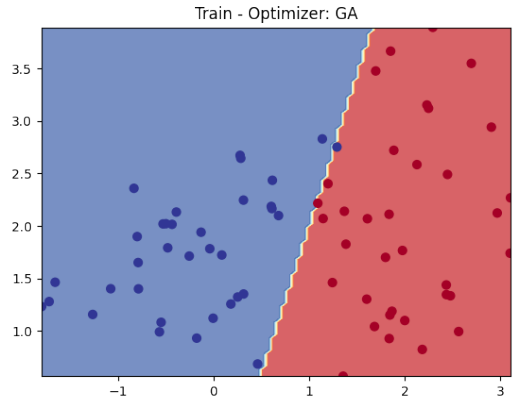
Figure 2.2: Decision boundary provided by the different algorithms over the **Classes** dataset

better generalisation across diverse instances. This difference in smoothness points to the interplay between optimisation algorithms and their ability to produce decision boundaries that balance accuracy and generalisation.

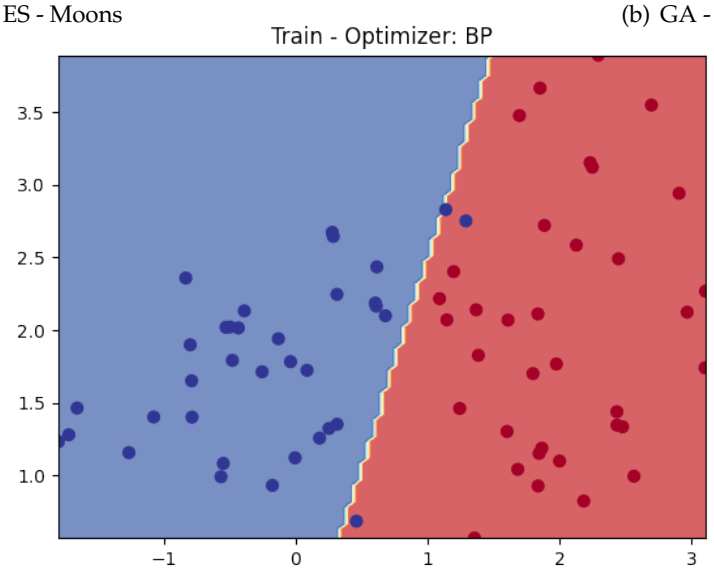
Furthermore, the decision boundaries resulting from the Stochastic Gradient Descent (SGD) optimisation method often showed remarkable accuracy. Although the bounds obtained with the SGD method show a precise delimitation of the regions within the feature space, this increased accuracy



(a) ES - Moons



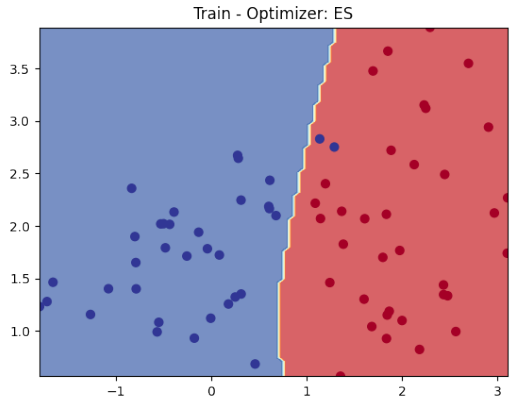
(b) GA - Moons



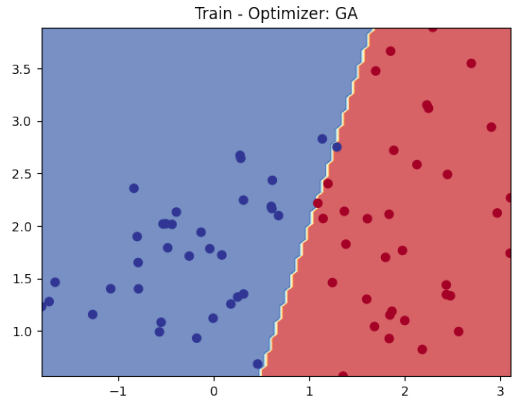
(c) SGD - Moons

Figure 2.3: Decision boundary provided by the different algorithms over the **Moons** dataset

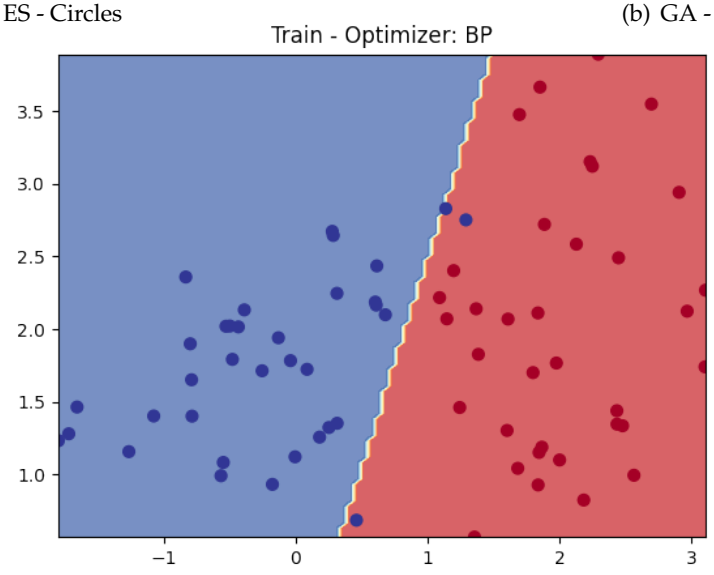
could pose problems of overfitting. The risk of overfitting arises when the model over-fits the training data, capturing noise as if it were a genuine pattern. Therefore, the accuracy of the decision boundaries derived from the SGD forces one to consider the delicate balance between the model's accuracy on the training data and its ability to generalise effectively to new and unknown data. This finding underscores the importance of hyperparameter fitting and regularisation techniques to achieve optimal balance and mitigate the challenge of overfitting associated with highly accurate decision boundaries.



(a) ES - Circles



(b) GA - Circles



(c) SGD - Circles

Figure 2.4: Decision boundary provided by the different algorithms over the **Circles** dataset

5 | Conclusions and future work

In conclusion, our experiments match expectations and highlight the superiority of derivative-based optimization methods, such as Stochastic Gradient Descent (SGD) coupled to Backpropagation, in terms of computational time. In particular, these methods often outperformed Evolution Strategies (ES) and Genetic Algorithms (GA) not only in computational efficiency but also, in certain cases, in terms of error and accuracy. The advantages of SGD + Backprop are evident in its fast convergence and efficient loss function minimization. However, it is crucial to recognize that ES and GA, although notably slower, present unique strengths that could be advantageous in specific scenarios. Evolutionary algorithms are inherently versatile and capable of handling noncontinuous or nondifferentiable objective functions, making them suitable for a wider range of optimization tasks. Moreover, their exploration-exploitation dynamics can be especially beneficial in complex, high-dimensional spaces, where traditional gradient-based methods may struggle.

Looking ahead, future work could explore the specific contexts in which ES and GA excel, potentially taking advantage of their ability to escape local minima and adapt to intricate and dynamic search domains. The adaptability of these evolutionary algorithms could prove advantageous in scenarios where one does not have a specific reward function. Another interesting strategy to explore, given the strengths of SGD + Backprop and evolutionary algorithms, may be to implement a hybrid system to try to combine the best of both worlds, combining the fast convergence of gradient-based methods with the robust adaptability of evolutionary strategies. By understanding the advantages of each optimization technique, could be easier to designm models adapted to the specific characteristics of different problem domains, which will increase the effectiveness of optimization algorithms in real-world scenarios.