

Global and Multi-Objective Optimization

Hyperparameters Optimization for Neural Networks with
Evolution Strategies

Gabriele Pintus

October 31, 2024

University of Trieste

Introduction

Introduction - Problem formalization

We want to approximate a function $f(x)$ with a neural network $\tilde{f}(x)$.

Let g be the function that maps the hyperparameters space \mathcal{H} to the output of a loss function, like \mathbb{R}^+ .

Then we want to solve:

$$h^* = \arg \min_{h \in \mathcal{H}} g(h)$$

In our case we will minimize the loss (MSE) on the validation set.

Introduction - The model

We choose our model to be a multi layer perceptron.

Moreover we will optimize the following hyperparameters:

- number of layers
- number of neurons per layer
- activation function
- dropout rate

As a consequence we define:

$$\mathcal{H} = \mathbb{N}_{1,5} \times \mathbb{N}_{16,128} \times \mathbb{N}_{1,6} \times \mathbb{R}_{0,0.5}$$

where $E_{a,b} = E \cap [a, b]$.

Genetic Algorithm

Genetic Algorithm

Genetic Algorithm (GA) is an optimization technique inspired by the process of natural selection and genetics.

Used to approximate solutions of optimization problems.

GAs work by:

- Generating a population of potential solutions.
- Applying evolutionary operators such as **selection**, **crossover**, and **mutation** to create new generations.
- Iterating until convergence criteria are met, evolving solutions toward an optimal or near-optimal solution.

We set:

- **population size:** 5
- **max generations:** 20

In our case:

Genotype: vector of hyperparameters

Phenotype: Neural Network

Fitness: Mean Squared Error on validation set

Genetic Algorithm - Operators

We decide to use the following operators:

- **Selection:** tournament selection of size $\tau = 3$, “mid pressure”
- **Crossover:** two-points crossover
- **Mutation:** random perturbation with distribution
 - number of layers: $\text{Unif}(-2, 2)$
 - neurons per layer: $\text{Unif}(-16, 16)$
 - activation function: ?
 - neurons per layer: $\mathcal{N}(0, 0.15)$

Since the activation function variable is categorical, mutating it is challenging. A trivial solution is to enumerate all activation functions and apply uniform mutation.

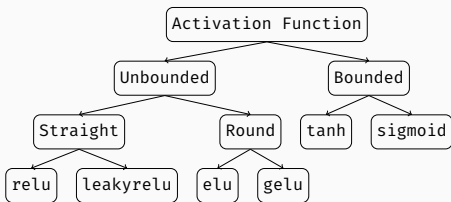
However, this approach implicitly assigns arbitrary distances between functions, leading to inconsistent outcomes with different enumerations.

Genetic Algorithm - Mutation operator

One better way to define the mutation operator, is to directly define a distance function between activation functions, and then induce a probability distribution.

Then, we can sample from this distribution to get the new activation function.

We begin by building a hierarchy of activation functions and represent it as a tree. Our choice is the following:



Genetic Algorithm - Mutation operator

We then define the distance function has the logarithm of number of the number of hops it takes to transition from one leaf to another.

$$d_b(l_1, l_2) = \log_b (\text{hops}(l_1, l_2))$$

Then we can induce a probability function

$$\begin{aligned}\tilde{p}_b(l_1, l_2) &= 2^{-d_b(l_1, l_2)} \\ p_b(l_1, l_2) &= \frac{\tilde{p}_b(l_1, l_2)}{\sum_{l \in L} \tilde{p}_b(l_1, l)}\end{aligned}$$

Varying the base b allow us to control the exploration-exploitation tradeoff. The higher the more explorative.

Implementation

Implementation

The algorithm has been implemented in Python. Along the usual machine learning libraries, two more frameworks have been used



Weights and Biases is a machine learning platform that enables tracking, versioning, and visualizing machine learning experiments.

PyTorch Lightning is a high-level framework built on top of PyTorch that simplifies the process of building and training deep learning models.

Every fitness assessment is extremely expensive as it involves the training of a neural network.

However, since feature assessments are mutually independent, the algorithm is inherently easy to parallelize. Moreover one can opt for two main parallelization strategies:

- **Population level:** Train all individuals at the same time, each using a portion of the total resources.
- **Individual level:** Train each individual sequentially, utilizing all resources for one at a time

PyTorch Lightning has already implemented several strategies which falls under the second case.

Parallelism

In large scale machine learning projects, the computational resources needed for the hp tuning process may be geographically far from each other.

Transferring data across them is time expensive. Think about Google training a model using resources distributed among multiple datacenters.

Parallelism

In large scale machine learning projects, the computational resources needed for the hp tuning process may be geographically far from each other.

Transferring data across them is time expensive. Think about Google training a model using resources distributed among multiple datacenters.

How to adapt a genetic/evolutionary algorithm?

Parallelism

In large scale machine learning projects, the computational resources needed for the hp tuning process may be geographically far from each other.

Transferring data across them is time expensive. Think about Google training a model using resources distributed among multiple datacenters.

How to adapt a genetic/evolutionary algorithm?

Island model → multiple populations coevolution

Results

Results

We tested both our GA algorithm and the standard Bayesian Optimization.

We use the Multi Layer Perceptron as neural network and two datasets:

- **Sinewave**: custom toy dataset. Five sinewaves as covariates, random linear combination as target.
- **MNIST**: handwritten digits.

Results - Sinewave

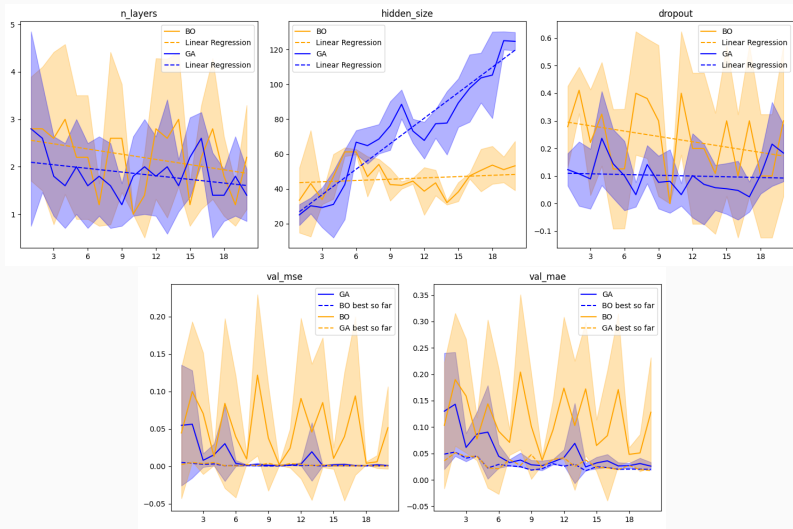


Figure 1: Hyperparameters evolution (top) and metrics (bottom)

Results - Sinewave

Five best models

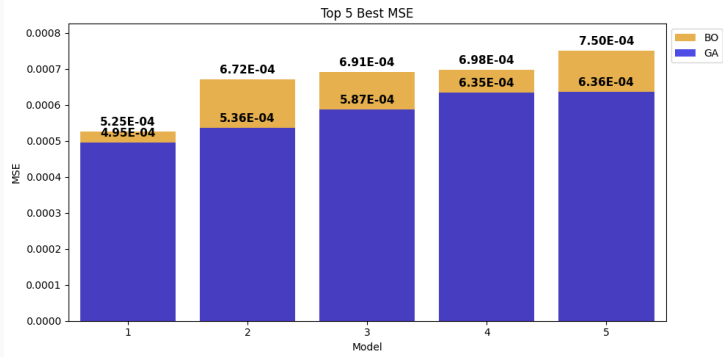


Figure 2: Hyperparameters evolution (top) and metrics (bottom)

Results - Sinewave

Top ten best models























| <input type="checkbox"/>  Name (400 visualizac | dropout | hidden_size | n_layers | activation_ | val_mse  |
|---|----------|-------------|----------|-------------|---|
|   fitness-9-5 | 0.028037 | 89 | 1 | 1 | 0.00049476 |
|   BO-Iteration-98 | 0 | 53 | 1 | 2 | 0.00052531 |
|   fitness-14-2 | 0 | 85 | 2 | 0 | 0.0005355 |
|   fitness-18-5 | 0.13645 | 128 | 1 | 1 | 0.00058669 |
|   fitness-10-4 | 0 | 84 | 2 | 1 | 0.00063492 |
|   fitness-17-1 | 0 | 92 | 1 | 0 | 0.00063611 |
|   fitness-19-2 | 0.14259 | 128 | 1 | 1 | 0.00064294 |
|   BO-Iteration-29 | 0 | 63 | 1 | 3 | 0.00067155 |
|   fitness-14-4 | 0 | 106 | 2 | 1 | 0.00068999 |
|   BO-Iteration-85 | 0 | 63 | 2 | 2 | 0.00069146 |

Figure 3: Screenshot from WaB

Results - MNIST

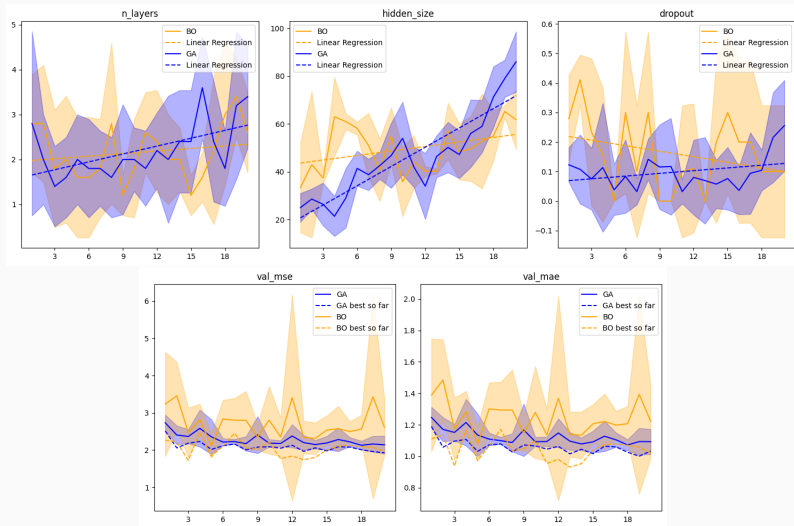


Figure 4: Hyperparameters evolution (top) and metrics (bottom)

Five best models

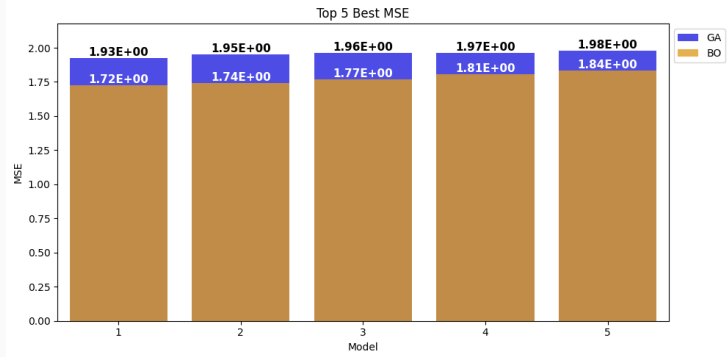


Figure 5: Hyperparameters evolution (top) and metrics (bottom)

Results - MNIST

Top ten best models
























| <input type="checkbox"/>  Name (200 visualizations) | dropout | hidden_size | n_layers | activation_ | val_mse  |
|--|----------|-------------|----------|-------------|---|
|   BO-Iteration-12 | 0.027062 | 36 | 1 | 4 | 1.72433 |
|   BO-Iteration-61 | 0 | 41 | 1 | 4 | 1.73948 |
|   BO-Iteration-55 | 0 | 41 | 2 | 4 | 1.7676 |
|   BO-Iteration-68 | 0 | 36 | 2 | 4 | 1.80567 |
|   BO-Iteration-24 | 0 | 59 | 1 | 4 | 1.83597 |
|   BO-Iteration-60 | 0 | 40 | 2 | 4 | 1.83981 |
|   fitness-20-2 | 0.25215 | 70 | 4 | 1 | 1.92628 |
|   fitness-20-1 | 0.13207 | 100 | 5 | 1 | 1.95205 |
|   fitness-19-5 | 0 | 88 | 4 | 0 | 1.96001 |
| <input type="checkbox"/>   BO-Iteration-96 | 0 | 67 | 3 | 4 | 1.96039  |

Figure 6: Screenshot from WaB

- More benchmark models and datasets
- More advanced distributed computing
- Adaptive exploration-exploitation mechanisms
- Variable length chromosomes
- Geometric Semantic Genetic Programming