Hyperparameter optimization using evolutionary algorithms for ML in high energy physics

Laurits Tani

National Institute Of Chemical Physics And Biophysics (NICPB), Akadeemia tee 23, 12618 Tallinn, Estonia

E-mail: `laurits.tani@cern.ch`

Abstract. In contemporary high energy physics (HEP) experiments the analysis of vast amounts of data represents a major challenge. In order to overcome this challenge various machine learning (ML) methods are employed. A complication arises from the fact that in addition to the choice of the ML algorithm a multitude of algorithm-specific parameters, referred to as hyperparameters, need to be specified in practical applications of ML methods. The optimization of these hyperparameters, which is often performed manually, has a significant impact on the performance of the ML algorithm. In this work we explore two distinct evolutionary algorithms, particle swarm optimization and Bayesian optimization, which allow to determine optimal hyperparameters for a given ML task in a fully automated way. We also studied the capability of these algorithms for utilizing the highly parallel computing architecture that is typical for the field of HEP.

1. Introduction

The majority of the present day high energy physics data analysis employ some machine learning methods in some shape or form. However, all of these algorithms feature a set of tunable parameters, called hyperparameters, that need to be chosen by the user prior to the training of the ML model. Depending on the ML algorithm, the choice of these hyperparameters can have a significant impact on the performance of the trained model. The strategy of choosing the optimal set of these hyperparameters, however, is not trivial and is done to this day often manually. Tuning hyperparameters manually, requires expert knowledge of the hyperparameters as well as the training data, consumes a lot of human time and makes the experiments not repeatable, motivating an automatized hyperparameter search. In addition to performing well the algorithms should be able to make full use of the high performance computing (HPC) cluster. For these purposes the comparison of two algorithms, particle swarm optimization (PSO) and Bayesian optimization (BO), featured in [1, 2] was made.

2. Particle swarm optimization

Particle swarm optimization (PSO) is an evolutionary algorithm that evolves a "swarm of particles", that with each iteration propagates through the hyperparameter space $\mathcal{H}$. The position of each particle in the swarm represents a set of hyperparameters $h \in \mathcal{H}$. There are in total $N_{parallel}$ particles in the swarm, which allows the user to train the same amount of ML models in parallel at each iteration of PSO.

The evolution starts with initializing $N_{parallel}$ particles in the hyperparameter space such that the swarm covers the hyperparameter space evenly. For the purposes of initializing the particles in the swarm we employed the *latin hypercube* [3] strategy.

The main part of the algorithm is the evolutionary loop, that is repeated until a certain criterion such as maximum number of iterations is reached. This evolutionary loop consists out of three main parts: *espionage*, *position update* and *speed update*. In the espionage step a subset of the particles in the swarm with a size of N_{info} is chosen randomly at each iteration k and per each particle i. This subset of particles is queried for their previously found best location, thus each particle i can update it's current best known location with respect to the objective function (OF) in $\mathcal{H}$.

In the next step of the evolutionary loop the position of each particle i is updated and evaluated. The position is updated according to $\vec{x}_i^{k+1} = \vec{x}_i^k + w \cdot \vec{p}_i^k + \vec{F}_i^k$, where $\vec{x}_i^k$ denotes the current position of the particle and $\vec{p}_i^k$ its momentum. The momentum term $w \cdot \vec{p}_i^k$ is prompting particles to prefer moving in the current direction. The symbol $\vec{F}_i^k$ represents an "attractive force", which compels particles to move towards previously discovered minima and is defined as:

$$\vec{F}_i^k = c_1 \cdot r_1 \cdot (\hat{\vec{x}}_i^k - \vec{x}_i^k) + c_2 \cdot r_2 \cdot (\hat{\hat{\vec{x}}}^k - \vec{x}_i^k)\,, \tag{1}$$

In the third and final step of the evolutionary loop the momentum of the particle is updated according to:

$$\vec{p}_i^{k+1} = \vec{x}_i^{k+1} - \vec{x}_i^k \tag{2}$$

In our implementation parameters for PSO are chosen to be $c_1 = c_2 = 1.62$, $w_{min} = 0.4$, and $w_{max} = 0.8$, as suggested in Ref. [4]. The parameters r_1 and r_2 in Eq.1 are random numbers that are drawn from a uniform distribution withing the interval of [0, 1].

3. Bayesian optimization

The second algorithm, the Bayesian optimization (BO) algorithm, is aimed at optimizing objective functions $s(h)$ that are expensive to evaluate and for which neither the analytic form nor derivatives are known.

This is made possible by the fact that BO does not perform the maximization on the $s(h)$ directly, but on an approximation of it which is referred to as the "surrogate" function (SF). The SF is chosen such that it is fast to evaluate and its analytic form together with the derivatives is known and that provides two values for each point $h \in \mathcal{H}$: the estimated value of s(h) and the uncertainty associated with that value.

Similarly to PSO the initial points in $\mathcal{H}$ are chosen according to the latin hypercube strategy.

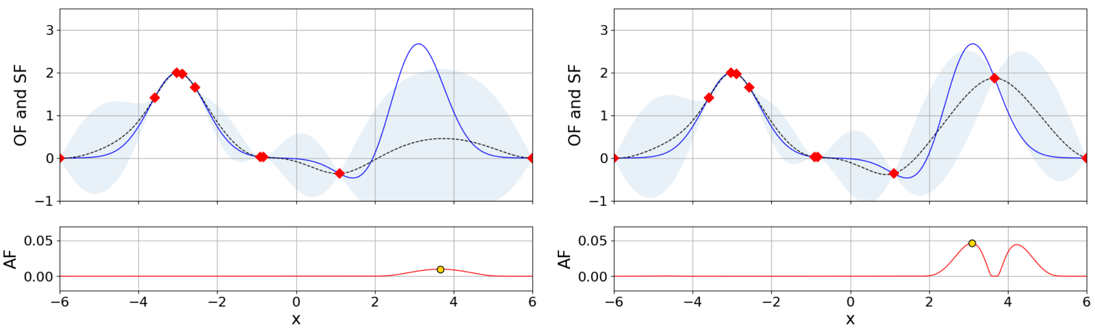
The optimization of the $s(h)$ is done iteratively, where each iteration consists of two main parts: finding the points h to evaluate and performing the time consuming evaluation of the OF at these points; and updating the SF with the newly found $s(h)$ values at the points h.

By default BO is a sequential algorithm. However in order to fully utilize the HPC , we follow the approach described in Ref. [5], referred to as "multi-points expected improvement" (q-EI), and use the implementation provided by Ref. [6] when running the hyperparameter parameter optimization on multiple machines in parallel.

Bayesian optimization is reported to work best with less than 1000 evaluations [7].

Fig. 1 shows the optimization process at iterations 7 and 8 respectively. The solid blue line in the upper part of each figure represents the objective function $s(h)$, dashed black line the SF, shaded blue area the confidence interval around the mean and the red diamonds the locations where $s(h)$ has been previously evaluated. On the lower part of each figure the solid red line denotes the acquisition function (AF) and the yellow circle along the line the next location where objective function $s(h)$ will be probed.

Figure 1. Iteration 7 (left) and 8 (right) of the BO algorithm, approximating the function $3\,e^{-(x-3)^2} - e^{-(x-2)^2} + 2\,e^{-(x+3)^2}$.



4. Benchmarks

In order to gauge the performances of PSO and BO two benchmark tasks, Rosenbrock function and ATLAS Higgs boson machine learning challenge, were chosen and are described in more details in the following.

4.1. Rosenbrock function

The Rosenbrock function [8] is a widely used trial function for evaluating the performance of function minimization algorithms and is defined as $R(x, y) = (a - x)^2 + b(y - x^2)^2$.

The Rosenbrock function is used directly as the objective function. The hyperparameters were optimized in the range of $(x \times y) = ([-500, +500] \times [-500, +500])$

4.2. ATLAS Higgs boson machine learning challenge (HBC)

In contrast to the easier, smooth, 2D benchmark task of the Rosenbrock function, the ATLAS Higgs boson machine learning challenge (HBC) [9] is more representative task for ML in HEP.

The goal of this challenge is to train a classifier that separates the events where Higgs boson is decaying to two taus ($\mathrm{H} \rightarrow \tau\tau$) from the 3 selected background categories:

- $\mathrm{Z} \rightarrow \tau_h\tau_h$ - decay of Z boson to two taus, which has a very similar topology to that of decay of Higgs to two taus
- $\mathrm{t\bar{t}} \rightarrow \tau_h + \mu/\mathrm{e}$ - pair of top quarks which might have a a lepton and a hadronic tau among the decay products
- W boson decay - electron or muon + hadronic tau can appear simultaneously only through imperfections in the particle identification process.

The classifier we trained was a boosted decision tree, XGBoost [10] in particular, for which we chose the seven hyperparameters shown in the Tab.1 to optimize, thus making the HBC in our context a 7 dimensional hyperparameter optimization problem.

Parameter	min	max	Parameter	min	max
num-boost-round	1	500	min-child-weight	0.0	500.0
learning-rate	10^{-5}	1.0	subsample	0.8	1.0
max-depth	1	6	colsample-bytree	0.3	1.0
gamma	0.0	5.0			

Table 1. The seven XGBoost hyperparameters that were optimized for the HBC task. Within the optimization, the hyperparameters were required to stay within the range indicated by the min and max columns.

The objective was to maximize the significance of the analysis, and for these purposes an objective function, approximate mean significance (AMS), defined by Eq.(3), was given by the organizers.

$$AMS = \sqrt{2 \cdot (s + b + b_r) \cdot ln[1 + \frac{s}{b + b_r}] - s} \tag{3}$$

As AMS depends only on the number of selected events which is a small part of the total events, it is prone to overfitting. Therefore we used the modified mean significance (dAMS), defined by Eq.(4), where we can have a handle on the overtraining through the parameter κ such that higher κ corresponds to less overtraining.

$$dAMS = AMS_{test} - \kappa \cdot max(0, [AMS_{test} - AMS_{train}]) \tag{4}$$

5. Results

Both Bayesian optimization and particle swarm optimization were ran for 30 iterations having 100 solutions evaluated concurrently. The performances of the two algorithms for Rosenbrock and HBC are shown on the left and right side of Fig.2 respectively. The public and private datasets of the HBC shown on right hand plot are the two testing datasets provided by the HBC organizers. As expected, Bayesian optimization performs in both cases better until $\sim 10^3$ evaluations, corresponding to the iteration 10 on Fig.2. Past this point particle swarm optimization passes Bayesian optimization in performance.

While the d-AMS score keeps increasing monotonously with more iterations, the AMS scores on the public and private leader board samples reach a plateau already after 5–10 iterations. In subsequent training iterations, the AMS scores start to fluctuate around the plateau. The magnitude of the fluctuations is of $\mathcal{O}(10^{-2})$. The differences in performance between the public and private leaderboard samples is compatible with a statistical fluctuation of the signal and background events contained in these samples [9]. The BO and PSO algorithms achieve a similar performance on the HBC task.

The ability to scale well with a high degree of parallelization is evaluated by calculating the speedup as per "Amdahl's law" [11]. The speedup on Fig.3 is shown relative to the sequential case. An ideal algorithm with negligible overhead would achieve a speedup factor equal to $N_{parallel}$ corresponding to a diagonal on Fig.3. As we can see then PSO is near ideal in that respect, taking only 0.01 CPUs per iteration for the 100 particles, whereas the evaluation of the Rosenbrock function takes 0.06 CPUs on average.

As the evaluation of the HBC is only $\mathcal{O}(30)$ CPU minutes, the Bayesian overhead is negligible for ML tasks that take hours, days or even weeks to evaluate.

With PSO being inherently a parallel algorithm, the optimal degree of parallelization is not a priori known in contrast to BO, which works best when run sequentially.

For the purposes of finding this optimal degree of parallelization the Rosenbrock function optimization was run for a multitude of different number of total evaluations and different degrees of parallelization.

As we can see in Fig.4 up to 10^4 total evaluations the optimal number of particles in the swarm was roughly constant - setting the number of particles in the swarm to a value of 2% times the total number of function evaluations and evolving the PSO algorithm for 50 iterations seems to work very well.

6. Summary

We compared two distinct autonomous algorithms, particle swarm optimization and Bayesian optimization, for the purposes of hyperparameter optimization on two benchmark tasks - the Rosenbrock function and the ATLAS Higgs boson machine learning challenge. For both

Figure 2. Bayesian optimization performs better than particle swarm optimization until the iteration 10 for both Rosenbrock function (left) and HBC (right) optimization tasks.

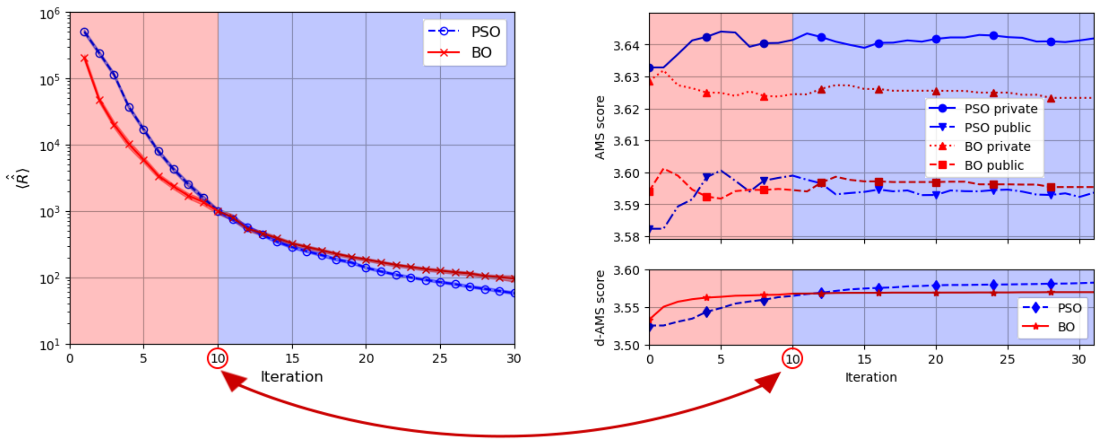
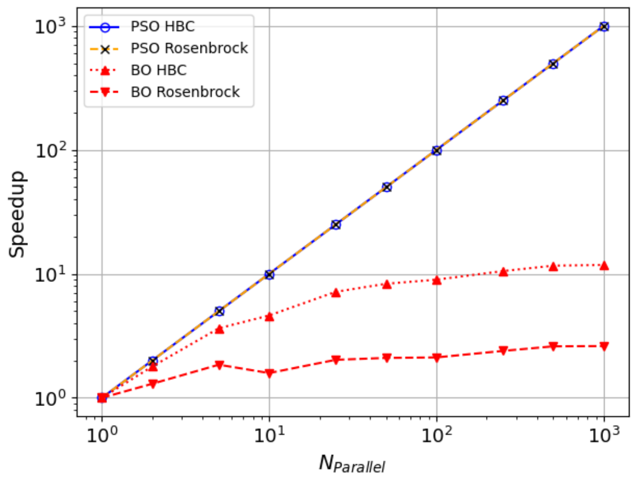


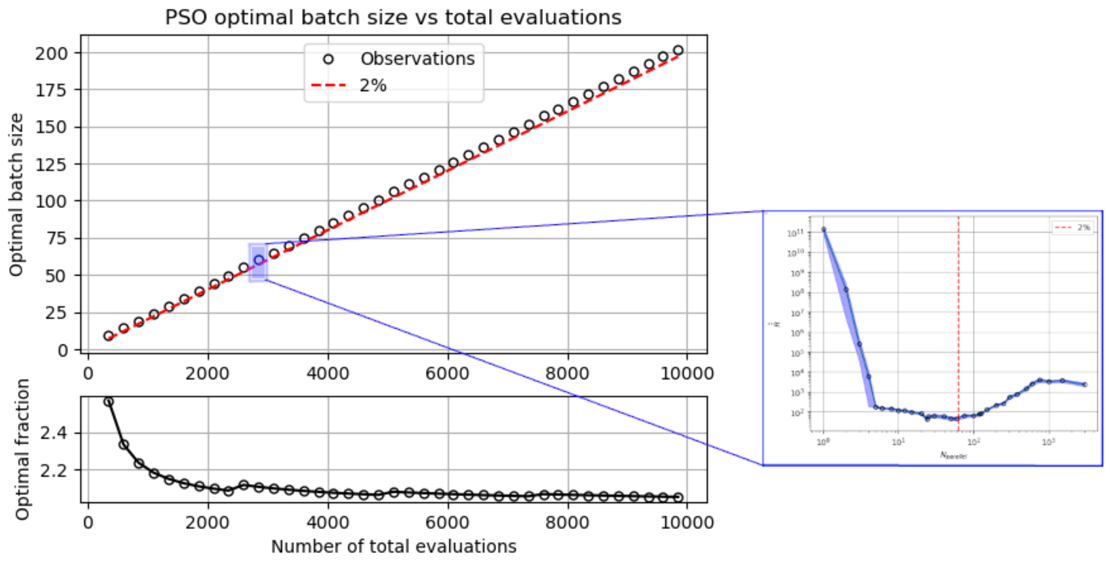
Figure 3. Amdahl's law: Parallelization properties of the BO and PSO algorithms on the two benchmark tasks. The two curves for the PSO algorithm are very close to the optimal scaling behavior, given by speedup $\approx N_{parallel}$.



benchmark tasks the Bayesian optimization converges faster up to $\sim 10^3$ total evaluation, after which the particle swarm algorithm passes the Bayesian one in performance. Therefore BO is more suitable in cases, where computational resources are limited. Both algorithms parallelize well, despite BO having some noticeable computational overhead for optimization tasks with fast to evaluate objective functions. PSO seems to work very well when setting 2% of the total number function evaluations as the swarm size and evolving the swarm for 50 iterations.

Acknowledgments
We would like to thank the ETAIS high-performance computing center at NICPB for granting us access to the computing resources necessary for this research. This work has been supported by the Estonian Research Council grant PRG445.

Figure 4. Left: On the top panel the optimal degree of parallelization is denoted by the empty circles and the 2% of the total evaluations is shown by the dashed red line. On the bottom panel the optimal relative degree of parallelization is shown as the solid black line. Right: the case of 3000 evaluations is shown, with the dashed red vertical line being at $60 = 3000 \cdot 2\%$, the solid black circles denoting the average found optimal Rosenbrock function value and the shaded blue area the uncertainty around the mean.



References

[1] L. Tani, D. Rand, C. Veelken, and M. Kadastik, "Evolutionary algorithms for hyperparameter optimization in machine learning for application in high energy physics," *The European Physical Journal C*, vol. 81, no. 2, pp. 1–9, 2021.

[2] L. Tani and C. Veelken, "Comparison of Bayesian and particle swarm algorithms for hyperparameter optimisation in machine learning applications in high energy physics," *arXiv preprint arXiv:2201.06809*, 2022.

[3] M. D. McKay, R. J. Beckman, and W. J. Conover, "A comparison of three methods for selecting values of input variables in the analysis of output from a computer code," *Technometrics*, vol. 21, no. 2, p. 239, 1979.

[4] M. Clerc, *Particle swarm optimization*, vol. 93. John Wiley & Sons, 2010.

[5] D. Ginsbourger, R. L. Riche, and L. Carraro, "A multi-points criterion for deterministic parallel global optimization based on Gaussian processes," tech. rep., HAL open science, hal-00260579, 2008.

[6] J. Wang, S. C. Clark, E. Liu, and P. I. Frazier, "Parallel Bayesian global optimization of expensive functions," *Oper. Res.*, vol. 68, p. 1850, 2020.

[7] P. I. Frazier, "A tutorial on Bayesian optimization," *arXiv:1807.02811*, 2018.

[8] Y.-W. Shang and Y.-H. Qiu, "A note on the extended Rosenbrock function," *Evolutionary Computation*, vol. 14, no. 1, p. 119, 2006.

[9] ATLAS collaboration, "Dataset from the ATLAS Higgs boson machine learning challenge," in *CERN Open Data Portal*, 2014.

[10] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proceedings, 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, p. 785, 2016.

[11] G. M. Amdahl, "Validity of the single processor approach to achieving large scale computing capabilities," in *Proceedings of the April 18-20, 1967, Spring Joint Computer Conference*, AFIPS '67 (Spring), (New York, NY, USA), p. 483, Association for Computing Machinery, 1967.