Surrogate Methods Applied to Hyperparameter Optimization Problem

José Ilmar Cruz Freire Neto[1] **e André Britto**[1]

[1] Departamento de Computação
Universidade Federal de Sergipe (UFS) – São Cristóvão, SE – Brazil

andre@dcomp.ufs.br, khoseilmar@gmail.com

Abstract. *Hyperparameters affects the performance of machine learning models. Hyperparameter optimization is an area that aims to find the best of them, but it deals with a considerable number of machine learning training, which can be slow. Thus, surrogates can be used to soften this slow process. This paper evaluates the performance of two surrogate methods, M1 and MARSAOP, applied to hyperparameter optimization. The surrogates are confronted with six hyperparameter optimization algorithms from the literature for classification and regression problems. Results indicate that the surrogate methods are faster than the traditional algorithms.*

1. Introduction

Machine learning models are capable of acting as a function of a problem by only observing some of its data. It trains using the data and can then predict the output given new data [Bishop and Nasrabadi 2006]. This can be useful when it is hard or impossible to define a function. So much that these models are widely used in the industry in lots of fields [Jordan and Mitchell 2015]. A problem is classified by the output of its data. When the output is categorical, the problem is a classification one and, when it is numerical, the problem is a regression one [Goodfellow et al. 2016]. This is important, because some machine learning models are better suited to a type of problem, since the strategies used in modeling a continuous function can not be applied or give poor results where the data has only two outputs, for example.

Some common machine learning models include: Linear Regression, K-Means and the SVMs. They all differ in the methods they use to train and make predictions. Linear Regression is a regression model that uses the training data to define a set of weights that will be used on the predictions. K-Means, on the other hand, creates groups for each categorical value and makes predictions based on the distance of the nearest groups. Finally, SVMs are a set of models based on the division of the space by a hyperplane. They are the models used in this paper and have both classification and regression models (in SVR case).

The models mentioned have parameters that affects how they train and predict. These parameters are called hyperparameters and must be defined before using the machine learning model [Yang and Shami 2020]. Therefore, a new problem appears. Besides choosing the suitable model for a problem, one must also choose the right hyperparameters. Optimization of hyperparameters is often used to overcome this problem [Bergstra et al. 2011].

Many optimization techniques are applied to the search for the best hyperparameters. As example, there are Grid Search, Random [Bergstra et al. 2011]Search and PSO [Poli et al. 2007]. Grid Search tries almost all the possible values, skipping some following a fixed pattern. Random Search tries a amount of random values. On the other hand, PSO defines a set of solutions called population and changes them, trying to get them closer to the current best solution.

The problem is modeled using the hyperparameters as decision variables and some metrics obtained after the model is trained as the objective function, such as the error rate of the machine learning model. Optimization techniques applied to the hyperparameters optimization problem will try to minimize/maximize the objective function and thus generate a better model.

The optimization will solve the hyperparameters issue. However, it is costly to train the machine learning model for every hyperparameter combination selected by the algorithm. The automatic parameter configuration problem can be defined as an Expensive Optimization Problem (EOP) [Tenne and Goh 2010]. EOPs are defined as problems where an objective function cannot be defined in an algebraic form, calculations of the objective function are computationally expensive or the problem has few feasible solutions. Some approaches, like parallel computing, the use of a surrogate, and the similarity of neighbors, have been proposed to solve these problems.

In the context of hyperparameter optimization, a surrogate can be seen as a simple and less costly function capable of calculate an approximation of the hyperparameter quality without training the machine learning model. Thus, the optimization algorithm can use a cheaper objective function skipping the mandatory training step for every new hyperparameter found.

So, these research questions were made: Is using surrogates in hyperparameter optimization competitive with the state-of-the-art techniques? Is it possible to obtain a better result in terms of execution time without losing the quality of the model?

This paper aims to evaluate the performance of surrogate techniques on hyperparameter optimization. The machine learning model used is SVM. Here, two surrogate methods from the literature were used M1 [**?**] and MARSAOP [Li et al. 2019]. M1 was applied initially only to benchmark optimization functions and obtained good results, while MARSAOP was applied to hyperparameter optimization, but only on a regression problem. To answer the research questions, a set of experiments were performed on classification and regression problems. The surrogates were confronted with six hyperparameter optimization techniques from literature using the execution time and accuracy metrics as performance criteria. Furthermore, the scalability of the surrogate methods was analyzed.

This paper is organized as follows. Section 2 presents the theoretical background and related work. The surrogates methods used in the experiments are discussed in 3. Section 4 discusses the experiments, and, finally the conclusions are presented in Section 5.

2. Background

Optimization consists in finding the best solution to a problem. Depending on the problem, the best solution is the one which minimizes or maximizes the objective function. In recent years, several problems with high complexity degrees have arisen. Most of those are real-world problems and the difficulties faced by the algorithms are mainly due to some difficulties like the objective function can not be defined in an algebraic form, the calculation of objective functions is computationally expensive or the problem has few feasible solutions.

The second reason is common in problems in which there is an objective function that represents the problem well, but the calculation of that function requires a high computational capacity or a long time for calculating (e.g. simulation). The use of a surrogate is an approach to solve such problems. It is a technique used to build very simple and cost-effective models, which seeks to replicate the relationships that are observed when examples of a more complicated model or simulation are built [Ranftl et al. 2021].

In the optimization context, a surrogate learns the behavior of the objective function and can be used to predict the objective values of a given solution [Coello 2017]. It can be a machine learning algorithm that will be trained using $x = (x_1, ..., x_n)$ solutions and its respective fitness $f(x)$, calculated by the objective function.

Once trained, the surrogate will be applied in the place of the objective function, that is, given a x solution as input, the surrogate will return a $f(x)$ value. It is noteworthy that the surrogate corresponds to an approximation of the objective function, so the value predicted by it does not correspond to the actual value that would be returned by the real objective function.

2.1. Hyperparameter Optimization Problem

Hyperparameters optimization is an optimization problem where the decision variables x are the hyperparameters and the objective function $f(x)$ is a function that calculates the error of the machine learning model. Usually, this calculation is done by training the machine learning model using the hyperparameters x as input and a quality metric that obtained by the trained model as objective function.

On classification problems, the quality metric used, usually, is the accuracy P, which ranges from 0 to 1, where bigger values mean better solutions. So, the objective function of hyperparameter optimization of a classification machine learning model calculates $1 - P$. This way, bigger values of P will minimize the objective. On regression problems, if the MSE is used as quality metric, it can be the objective.

This way, hyperparameter optimization can be defined as:

$$x^* = argmin_{x \in X} f(x),$$

where x^* is the optimal hyperparameter found.

Since it is necessary to train the machine learning model to calculate the objective function, it is possible to say that one evaluation is costly. So, an hyperparameter optimization problem can be classified as an expensive optimization problem, which benefits from the use of surrogates.

2.2. Related Work

In [Yang and Shami 2020], 8 different optimization algorithms where applied in hyperparameter optimization. Each of them was compared in machine learning algorithms, including the SVM, using the datasets MNIST and Boston Housing, for classification and regression, respectively. In the SVM optimization, two algorithms stood out: BOHB and BO-TPE. They achieved the shortest time, lowest error rate and good accuracy compared to the others. It can be noted that, as in most state-of-the-art SVM hyperparameter optimization, the parameters optimized were C, kernel, and, only in the regression problem, epsilon.

Li et. al [Li et al. 2019] proposed the surrogate MARSAOP based on the machine learning algorithm MARS and applied it to the hyperparamenter optimization problem. Experiments using search algorithms and MARSAOP were executed in some benchmarking problems and in the dataset MNIST. Machine learning algorithms, including the SVM, were used optimizing the same hyperparameters of the previous work, C and kernel. The results showed that the MARSAOP has a good performance.

In [Grama et al. 2017], grid search using log2-space was applied in the hyperparameter optimization of an SVM classifier. The dataset used posses audio signals divided into 5 classes: birds, chainsaws, gunshots, human voice, and tractors. The parameters optimized was C, γ, and r, where r is only optimized when the kernel sigmoid is used. It is important to notice that three kernel parameters were used: linear, radial basis, and sigmoid, but not optimized. Instead, one of them was selected before the optimization started. In conclusion, compared to the default hyperparameters, the optimization improved the accuracy of the machine learning model, especially while using the sigmoid kernel.

A method to optimize SVM hyperparameters using pre-processed data was proposed in [Duo et al. 2017]. The paper explored two search algorithms: PSO and GPSO. The last one is a modified version of PSO that uses some concepts of evolutionary algorithms to avoid early convergence to local optimums. It was concluded that algorithm using the data pre-processing and the GPSO had the best results.

In [Kamath 2022] proposes the concept of intelligent sampling, where the authors devise solutions specifically tailored to meet a sampling needs. The results indicate that some simple algorithms can be easily modified to meet the many sampling requirements of surrogate modeling, hyperparameter optimization, and data analysis

In summary, most works in the state-of-the-art propose new algorithms applied to optimize SVM hyperparameters. Nonetheless, only the work presented in [Li et al. 2019] modeled the problem as an expensive optimization problem. MARSAOP results were promising, but the experiments were limited, and it was not tested on a regression problem. Furthermore, other surrogate methods, besides those based on bayesian optimization, in the literature are not evaluated in the hyperparameters optimization problem.

3. Surrogate Methods

In our work, two surrogate methods were used: M1 and MARSAOP. The first, M1, has the idea of improving surrogates with solutions found by an evolutionary algorithm. This algorithm can be applied for both mono and multi objective optimization. Its implemen-

tation and explanation can be found on the article that proposed it [Deb et al. 2019]. In short, the M1 receives an evolutionary algorithm and one or multiple objective functions, trains one surrogate for each objective function using a set of solutions and, by running the evolutionary algorithm, tries to find better solutions with the surrogates as objective functions. At first, the set of solutions is randomly generated. After the run of the evolutionary algorithm, its output is added to the set of solutions. Then, the M1 retrains the surrogates. These steps are repeated until the number of evaluations with the objective functions reaches a threshold defined as parameter.

The second implemented surrogate, the MARSAOP, has its own search algorithm and was made aiming to be used with the machine learning algorithm MARS. The MARSAOP's article [Li et al. 2019] offers its implementation and explanation. In short, the surrogate is about evaluating with the objective function, train a surrogate using MARS with the solutions and objectives, generate candidate points using the current solution set, select a evaluation point from the candidate points, put this evaluation point at the current solution set, update the variance value and repeat until some condition is met.

The generation of candidate points using the current solution set is done by selecting the best solution of this set and generating some solutions from it. Each one of these new solutions is a mutation of the best solution, because a new solution copy the variables of the best solution, selects some of these variables, randomly generating numbers between 0 and 1 for each variable and choosing the ones which the value is below a parameterized threshold, and apply a different gaussian noise, using the variance value, to each one selected.

The evaluation point is the one which have the smaller linear combination of all the candidate points. The terms of this linear combination are the smaller distance between the current solutions and the normalized distance between its objective calculated by MARS and the minimum objective calculated by MARS from the generated solutions. In contrast, the coefficients are parameters of the MARSAOP.

Lastly, the variance update is done when the objective of the best solution from the generated solutions was smaller or not than the objective of the current best solution a number of times. When the update is done, the count of times for both conditions, i. e., smaller or not, is reseted.

4. Experiments

Two experiments were performed to evaluate the performance of the surrogate methods applied hyperparameter optimization of SVM. The first experiment the surrogate algorithms are confronted to 6 hyperparameter optimization algorithms from literature, for both classification and regression problems. The second experiment analysis the scalability of the surrogate algorithms. For this experiment M1 and MARSAOP are confronted to PSO algorithm, varying the number of time where SVM is trained.

4.1. Dataset

In the classification experiment, the dataset UCI ML hand-written digits was used. It contains images of hand-written digits and is the same dataset used in the implementation given by [Li et al. 2019]. In total, there are 1797 images, where each one has a dimensionality of 64. In the regression experiment, the dataset Boston Housing was used. Different

Tabela 1. M1 Parameters

Algorithm	Number of evaluations	ρ	τ	SE_{max}	GA pop size	GA max evals
M1 25%	100	5	2	25	10	20
M1 50%	100	25	1	50	25	25
M1 75%	100	75	1	75	25	25
M1 25%	1000	50	1	250	50	150
M1 50%	1000	50	1	500	50	50
M1 75%	1000	750	1	750	250	250
M1 25%	10000	500	1	2500	500	1500
M1 50%	10000	500	1	5000	500	500
M1 75%	10000	7500	1	7500	2500	2500

from the UCI's one, this dataset has data from houses in the city of Boston. There are 506 houses with dimensionality equals to 14 in this dataset.

All datasets were split using stratified 3-fold cross validation.

4.2. Algorithms and parameters

For the first experiment, the hyperparameter optimization algorithms used were Grid Search, Random Search [Bergstra et al. 2011], BO-GP (bayesian optimization with gaussian process) [Seeger 2004], BO-TPE (bayesian optimization with tree Parzen estimator) [Hutter et al. 2011], PSO [Kennedy 2010] and GA [Lessmann et al. 2005]. Here, these algorithms will be referred as traditional algorithms. Of each one, it is possible to change the search space and the number of evaluations with the objective function. It is also possible to change other parameters, but they were ignored, keeping the predefined values from the libraries.

Regarding the search space, three parameters from SVM were considered: C, kernel, and epsilon. For the first, values from 0.1 to 50 were allowed. For the second, which is a categorical parameter, only the values linear, poly, rbf, and sigmoid were allowed. For the last, values in the real interval $[0, 1]$ were allowed. Note that, in the classification problem, the parameter epsilon and, in the regression problem, the kernel linear are not available.

For the second experiment, the number of evaluations was increased to observe how the surrogate methods perform. In this analysis, besides M1 and MARSAOP, only PSO was chosen since it had the best results in experiment 1. The main parameters remain the same. Only the number of evaluations was increased to 1000 and 10000.

The M1 was executed with a Genetic Algorithm using Polynomial Mutation as the mutation method and SBX as the crossover method. In Polynomial Mutation, the mutation distribution index was 20, and the probability was 0.1. In SBX, the distribution index was the same, but the probability was 0.9. The M1 was divided in three configurations: M1 25%, M1 50% and M1 75%, where each does 25%, 50% and 75%, respectively, evaluations using the objective function of the total evaluations. M1 and GA parameters are presented in table 1.

In MARSAOP, in both experiments, m equals to 3, σ equals to 0.2, ϕ_0 equals to 1.0, w_v equals to 0.81, w_u equals to 0.19, thr_1 equals to 7 and thr_2 equals to 7 was used.

Tabela 2. Comparison of all algorithms on a classification problem

Algorithm	Average accuracy (%)	Average time (s)	σ accuracy	σ time
Grid Search	97,38	13,52	0	0,04
Random Search	97,43	11,64	0	0,02
BO-GP	97,41	322,37	0	0,19
BO-TPE	97,39	3,24	0	0,53
PSO	97,41	2,45	0	0,14
GA	97,44	5,14	0	0,13
M1 25%	97,39	2,42	0	0,36
M1 50%	97,41	5,84	0	0,18
M1 75%	97,43	10,22	0	0,09
MARSAOP	97,38	2,79	0	0,57

The parameter G_{max} changes with the number of evaluations. So, he was 25, 250 and 2500, for 100, 1000 and 10000 evaluations, respectively.

For the execution of the mentioned algorithms, the following libraries were used: sklearn, scipy, skopt, hyperopt, optunity, evolutionary_search, pyDOE2 and pyearth. Detailing, it was used the implementations of Grid Search, Random Search, Linear Regression and SVM from sklearn, of BO-GP and BO-TPE from skopt, of PSO from optunity, of GA from evolutionary_search, of LHS from pyDOE2 and of MARS from pyearth.

4.3. Quality metrics

The accuracy and execution time were used to compare the performance of the algorithms in the classification problem. In total, 25 executions of each algorithm were made. So, each one has 25 accuracies and 25 execution time. So, the performance of each algorithm is obtained by calculating the mean of the 25 accuracies and the mean of the 25 execution time. The bigger the average accuracy, the greater the number of right predictions done by the SVM. Consequently, the better is the algorithm's quality. The smaller the average execution time, the faster the algorithm found a parameter.

The performance comparison in the regression problem is similar to the classification, but the accuracy is replaced by the MSE. The smaller the MSE, the closer the predicted results by the SVM are from the real results and, consequently, the better is the algorithm.

4.4. Results and discussion

The first experiment concerns the execution of all algorithms, using 100 evaluations. Table 2 presents the results of the classification problem, while Table 3 the results of regression problem.

For the classification problem, GA obtained greater accuracy, with little difference when compared to the other algorithms. When analyzing the execution time, the PSO was the fastest. The M1 25% configuration with the lower value of SE_{max} was almost as fast as PSO, in the second place overall, but with a greater standard deviation in time, compared to most algorithms. BO-GP was about 100 times worse than PSO. Finally, there is a 10 seconds interval between the other algorithms.

Tabela 3. Comparison of all algorithms on a regression problem

Algorithm	Average MSE	Average time (s)	σ MSE	σ time
Grid Search	60,15	3,7	0	0,02
Random Search	59,13	3,3	0,49	0,01
BO-GP	58,76	308,03	0	0,17
BO-TPE	58,7	0,77	0,48	0,02
PSO	59,34	1,09	0,39	0,11
GA	59,74	1,11	0,52	0,17
M1 25%	60,73	0,89	2,57	0,1
M1 50%	59,69	2,02	0,64	0,06
M1 75%	59,3	2,91	0,5	0,05
MARSAOP	62,37	1,04	3,52	0,13

Tabela 4. Scalability analysis with 1,000 evaluations for a classification problem

Algorithm	Average accuracy (%)	Average time (s)	σ accuracy	σ time
PSO	97,45	73	0	7,61
M1 25%	97,42	21,45	0	0,94
M1 50%	97,43	30,33	0	4,16
M1 75%	97,44	97,4	0	2,19
MARSAOP	97,38	24,57	0	7,3

In the regression problem, BO-GP and BO-TPE got the smallest error. In comparison to execution time, BO-TPE was the best. About the others, one can tell that, excluding BO-GP, everyone achieved a good time. Putting GA aside, it is also possible to conclude that the time are proportional to the classification problems.

As surrogates exchange quality for time, it is expected that they are faster than the other algorithms. Nevertheless, in the classification problem, PSO achieved a shorter time than the fastest surrogate, MARSAOP. It is fair to give credit to the small number of function evaluations, where the change of context to train a surrogate is more expensive than the own function evaluation.

The second experiment analyses the scalability of the surrogate methods. Here, it was observed how the performance of the surrogate methods varies when the number of evaluations grows. In this analysis, both M1 and MARSAOP were considered. Also, PSO, the traditional algorithm that obtained the best results in experiment 1, was used as a basis. Tables 4 and 5 presents the results of accuracy/MSE and execution time, for classification and regression, respectively, for $1,000$ objective function evaluations. Tables 6 and 7 present the results for $10,000$ evaluations.

For 1000 evaluations, in the classification problem, PSO got the best accuracy. The surrogates, on the other hand, did not get too far away. Moreover, it is possible to see that the accuracy values are very similar to those of the 100 evaluations, with a difference of, at max, 0.1%. So, given the greater number of evaluations, possibly, the result of the algorithm and the surrogates was better, even by little.

Regarding the time, the differences are more evident compared to the other expe-

Tabela 5. Scalability analysis with 1,000 evaluations for a regression problem

Algorithm	Average MSE	Average time (s)	σ MSE	σ time
PSO	58,97	43	0,5	6,41
M1 25%	58,84	9,38	0,41	0,51
M1 50%	59,1	9,035	0,51	0,56
M1 75%	58,48	28,64	0,14	0,33
MARSAOP	61,86	7,65	2,89	1,12

Tabela 6. Scalability analysis with 10,000 evaluations for a classification problem

Algorithm	Average accuracy (%)	Average time (s)	σ accuracy	σ time
PSO	97,49	669	0	93,71
M1 25%	97,44	206,64	0	4,78
M1 50%	97,44	305,47	0	54,29
M1 75%	97,44	1010,78	0	4,86
MARSAOP	97,38	198,2	0	11,39

riment, so much that it is possible to give better conclusions. For example, the M1 25% is faster than the others; this time, PSO had a worse result than MARSAOP, M1 25% and M1 50%. The surrogate that uses more evaluations with the objective function, M1 75%¨, had the worst execution time. This way, talking about the M1, in this experiment, evaluating more with the objective function is not worth it, because the accuracy gain was little, but the time spent was not. It is also possible to notice the high standard variation of MARSAOP and PSO.

In the regression problem, the M1 75% was the one with the lowest MSE and the MARSAOP with the biggest. Beyond that, all MSEs were lower. Concerning the time, the PSO was worse than all surrogates. The comparison of the M1s done in the classification problem showed to not be also true in this problem, since the M1 50% was executed is less time, but with a worse MSE.

The results of the $10,000$ evaluations in the classification problem were a bit different to the $1,000$ evaluations experiment. MARSAOP, this time, was the fastest, but still showed worse accuracy and bigger standard deviation. PSO continued to lose only to M1 75%. The conclusion of the advantage of the M1 25% in comparison to the other ones with more function evaluations proved to be true again.

In the regression problem, a peculiar fact can be seen. The PSO had a worse time than its time in the classification problem. Furthermore, M1 25% was the fastest. Note that all surrogates achieved MSE, excluding MARSAOP, and time lower than PSO.

5. Conclusions

In this paper a comparison between different hyperparameter optimization methods was done. This paper was focused on analysing the performance of the application of surrogates on hyperparameter optimization. The surrogate methods explored where M1 and MARSAOP.

The experiments used two datasets, UCI ML hand-written digits and Boston Housing, one of classification and the other of regression, respectively. M1 and MARSAOP

Tabela 7. Scalability analysis with 10,000 evaluations for a regression problem

Algorithm	Average MSE	Average time (s)	σ MSE	σ time
PSO	58,6	1219	0,33	182,09
M1 25%	58,61	72,47	0,05	0,66
M1 50%	58,54	94,76	0,01	0,43
M1 75%	58,29	291,88	0,01	1,73
MARSAOP	62,58	86,23	1,04	9,52

were confronted to 6 different traditional algorithms. Two set of experiments were conducted, the first one comparing all the algorithms, and the second one analyzing the scalabitliy of the surrogate methods.

With these experiments, the research questions can be answered. The surrogate methods are competitive with the traditional algorithm. Furthermore, it was possible to obtain faster results without losing quality in some scenarios. The surrogate methods showed to be faster than the popular search algorithms, especially, on tests with more evaluations while keeping almost the same quality. Future work includes: use neural networks as machine learning methods and compare with other surrogates methods and datasets.

Referências

Bergstra, J., Bardenet, R., Bengio, Y., and Kégl, B. (2011). Algorithms for hyper-parameter optimization. *Advances in neural information processing systems*, 24.

Bishop, C. M. and Nasrabadi, N. M. (2006). *Pattern recognition and machine learning*, volume 4. Springer.

Coello, C. A. C. (2017). Recent results and open problems in evolutionary multiobjective optimization. In *International Conference on Theory and Practice of Natural Computing*, pages 3–21. Springer.

Deb, K., Hussein, R., Roy, P. C., and Toscano-Pulido, G. (2019). A taxonomy for metamodeling frameworks for evolutionary multiobjective optimization. *IEEE Transactions on Evolutionary Computation*, 23(1):104–116.

Duo, M., Qi, Y., Lina, G., and Xu, E. (2017). A short-term traffic flow prediction model based on emd and gpso-svm. In *2017 IEEE 2nd Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*, pages 2554–2558. IEEE.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. `http://www.deeplearningbook.org`.

Grama, L., Tuns, L., and Rusu, C. (2017). On the optimization of svm kernel parameters for improving audio classification accuracy. In *2017 14th International Conference on Engineering of Modern Electric Systems (EMES)*, pages 224–227. IEEE.

Hutter, F., Hoos, H. H., and Leyton-Brown, K. (2011). Sequential model-based optimization for general algorithm configuration. In *International conference on learning and intelligent optimization*, pages 507–523. Springer.

Jordan, M. I. and Mitchell, T. M. (2015). Machine learning: Trends, perspectives, and prospects. *Science*, 349(6245):255–260.

Kamath, C. (2022). Intelligent sampling for surrogate modeling, hyperparameter optimization, and data analysis. *Machine Learning with Applications*, 9:100373.

Kennedy, J. (2010). *Particle Swarm Optimization*, pages 760–766. Springer US, Boston, MA.

Lessmann, S., Stahlbock, R., and Crone, S. F. (2005). Optimizing hyperparameters of support vector machines by genetic algorithms. In *IC-AI*, volume 74, page 82.

Li, Y., Liu, G., Lu, G., Jiao, L., Marturi, N., and Shang, R. (2019). Hyper-parameter optimization using mars surrogate for machine-learning algorithms. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 4(3):287–297.

Poli, R., Kennedy, J., and Blackwell, T. (2007). Particle swarm optimization. *Swarm intelligence*, 1(1):33–57.

Ranftl, S., von der Linden, W., and Committee, M. . S. (2021). Bayesian surrogate analysis and uncertainty propagation. In *Physical Sciences Forum*, volume 3, page 6. MDPI.

Seeger, M. (2004). Gaussian processes for machine learning. *International journal of neural systems*, 14(02):69–106.

Tenne, Y. and Goh, C.-K. (2010). *Computational Intelligence in Expensive Optimization Problems*. Springer Berlin, Heidelberg, 1 edition.

Yang, L. and Shami, A. (2020). On hyperparameter optimization of machine learning algorithms: Theory and practice. *Neurocomputing*, 415:295–316.