Time-Optimized Bayesian Optimization of Random Forest Hyperparamaters

Gabriel Xu and Andrew Chen

January 31, 2025

Machine Learning Quarter 2 Project

Dr. Yilmaz Period 5

Abstract

Finding optimal hyperparameters is crucial for machine learning models, yet it is both challenging and time-consuming. Traditional approaches, such as manual hyperparameter selection and grid search, rely heavily on trial and error and often produce inconsistent results. Bayesian Optimization has been proposed as a more efficient alternative, which leverages a probabilistic surrogate model to help guide hyperparameter selection. Nevertheless, existing approaches rely on the manual selection of Bayesian Optimization's hyperparameters itself, and such selection can introduce bias and cause inefficiencies. This study presents a novel approach that meta-optimizes the Bayesian Optimization process, allowing it to tune its own hyperparameters while optimizing the Random Forest Classifier. We apply this method to the breast-cancer.csv dataset from Weka. We compare three models: (1) a baseline Random Forest with manually chosen hyperparameters, (2) a Bayesian-optimized Random Forest with predefined settings, and (3) our meta-optimized approach, where Bayesian Optimization itself is tuned recursively. Performance is evaluated based on classification accuracy, precision, recall, training time, and confusion matrices. Preliminary results indicate that meta-optimization improves classification accuracy and reduces hyperparameter search time. Our findings suggest that automating Bayesian Optimization settings can enhance the performance and efficiency of machine learning classifiers. Future work may explore applying this approach to other machine learning models and different datasets.

Introduction

In this project, we address the decision trees and random forests classification hyperparameter optimization problem. Decision trees represent one of the most popular data mining techniques which have been widely used to solve classification problems; however, they are sensitive to the choice of their hyperparameters, such as tree depth and minimum samples per split. The task of tuning hyperparameters for such models is mostly difficult and either uses heuristics or hand-tuning methods, which again turn out to be inefficient and suboptimal. Bayesian optimization has been previously attempted research for this very problem, but current methods themselves use hand-specified hyperparameters and thus come with potential biases and inefficiencies. All the models we test in this project will be tested on the same dataset in controlled conditions to ensure a level playing field for all of them. We will check whether our proposed method

could significantly improve classification accuracy and efficiency. For our algorithm, the input will be the dataset detailed below and we will run random forest on the dataset, outputting predicted recurrence event or not, giving us metrics of accuracy, time it took to run, precision, recall, confusion matrix, etc. that we can use to compare our method with other ones.

Related Work

Conventional approaches for hyperparameter optimization, such as grid search and random search, can become inefficient when exhaustively searching through hyperparameters with no use of prior information. Bayesian Optimization, in contrast, has emerged as a newer and more efficient alternative that leverages a probabilistic surrogate model to guide the search process and identify optimal hyperparameters (Snoek et al., 2012).

One notable study, “Hyperparameter Optimization for Machine Learning Models Based on Bayesian Optimization” (Zhang et al., 2019), considers Bayesian Optimization in tuning machine learning models, using a Gaussian process (GP) model for approximating objective function and guiding hyperparameter search. According to the study, Bayesian Optimization maximizes search efficiency by systematically refining hyperparameter selection based on prior evaluations, therefore leading to more precise and effective model tuning. Nevertheless, this approach relies on manually specifying parameters specific to Bayesian Optimization, such as the number of initial sample points and the choice of the acquisition function, which as a result can introduce bias and restrict generalizability across different datasets and models.

To address this limitation, modern studies have focused on hyperparameter automation in Bayesian Optimization. For example, Garrido-Merchán & Jariego-Pérez (2021) propose dynamically optimizing acquisition functions to balance exploration and exploitation, reducing the need for manual intervention. However, their approach relies on predefined heuristics to determine when to switch between acquisition functions, which may not adapt well to all optimization problems. In a similar study, Lindauer et al. (2019) evaluate the impact of Bayesian Optimization’s inner hyperparameters on performance, concluding that tuning these parameters can improve convergence speed

and efficiency. However, this comes at a computational cost, making this approach impractical for larger datasets.

In addition to Bayesian Optimization, several alternatives for hyperparameter automatization have been developed. Auto-WEKA (Thornton et al., 2013) is one such algorithm; it automates both algorithm selection and hyperparameter optimization, allowing machine learning pipelines to be configured without manual intervention. This approach is practical, but it is computationally intensive and thus difficult to scale towards larger datasets.

Our work builds upon these insights and introduces a meta-optimization approach that recursively tunes the hyperparameters of Bayesian Optimization itself. By doing so, we hope to improve the accuracy and efficiency of such models and make hyperparameter tuning more adaptable to different datasets. By reducing the need for manual parameters and decreasing computational costs, our approach creates a more reliable and scalable way to optimize hyperparameters in machine learning models.

Dataset and Features

We used the breast-cancer.csv dataset found in Weka [7], which was taken from the UC Irvine Machine Learning Repository [8], as it is a good example of a problem that can be solved with the random forest algorithm. The dataset contains clinical and histopathological data collected from patients diagnosed with breast cancer. It includes various features such as tumor size, patient age, hormone receptor status, and other relevant medical attributes, making it useful for performance analysis in classification models predicting malignancy and treatment outcomes. For this study, the dataset was divided into training (80%) and testing (20%) sets, ensuring that class balance was maintained to prevent bias in model training.

Categorical features such as hormone receptor status were converted using one-hot encoding to become numbers that would work with the random forest classifier. Discretization was also applied to certain variables, which was already done to the dataset in Weka for the features “menopause”, “tumor-size”, and “inv-nodes”. Since random forest is a classification problem that works with categorical data there was no

need to perform normalization. We used all of the features from the dataset as none of them had too much missing data and were useful. Here are some example instances of the dataset:

40-49	premeno	15-19	0-2	yes	3	right	left_up	no	recurre...
50-59	ge40	15-19	0-2	no	1	right	central	no	no-rec...
50-59	ge40	35-39	0-2	no	2	left	left_low	no	recurre...
40-49	premeno	35-39	0-2	yes	3	right	left_low	yes	no-rec...
40-49	premeno	30-34	3-5	yes	2	left	right_up	no	recurre...
50-59	premeno	25-29	3-5	no	2	right	left_up	yes	no-rec...

Methods

Our approach tries to further enhance using the Bayesian optimization process on a random forest model by using Bayesian optimization not only for tuning the random forest but also for the optimization of Bayesian optimization algorithm hyperparameters. This optimization strategy might improve classification performance while making the process of hyperparameter tuning more systematic and less arbitrary. We will be comparing three methods: a vanilla random forest with hyperparameters selected by hand, a random forest optimized by Bayesian optimization using predefined settings, and our approach, in which Bayesian optimization is performed in a recursive manner.

The major machine learning algorithm used for this project is the Random Forest Classifier, an ensemble learning technique whereby several trees are constructed and the predictions of these trees are aggregated to give more accurate classifications while reducing overfitting. A decision tree is a hierarchical structure where every node represents a feature, and every branch a decision rule. One works one's way down the tree, starting at the root and ending at a leaf node to determine the case's class in question. To resolve overfitting, a random forest builds multiple decision trees and outputs the most common answer coming from those trees via majority voting. The most important hyperparameters of a random forest are the number of trees, maximum depth of a tree, minimum samples required to split a node, and minimum samples required to be at a leaf node. These greatly affect performance and therefore need to be optimized.

Bayesian Optimization is a probabilistic model-based approach to performing global optimization in an efficient manner, especially for expensive function evaluations. Unlike

grid search or random search, Bayesian optimization constructs a surrogate function-a Gaussian process-to approximate the unknown objective function and then chooses the next set of hyperparameters by balancing exploration and exploitation using an acquisition function. Formally, for an objective function $f(x)$, Bayesian optimization models it using a Gaussian process $GP(\mu(x),\ k(x,x'))$, where $\mu(x)$ is the mean function and $k(x,x'))$ is a kernel function defining similarity between points. The acquisition function, such as Expected Improvement (EI) or Upper Confidence Bound (UCB), selects the next set of hyperparameters $x\ *$ by maximizing $x\ *\ =\ arg\,max\,a(x)$ where $a(x)$ is the acquisition function. The hyperparameters of this algorithm that we can tune are $init\_points$ and n_iter, the initial number of points to test and the number of iterations of the algorithm. Higher number of both of these would be better, but we also need to consider how it takes more time, so our algorithm optimizes the accuracy and time it takes for the optimization to run.

The code for this can be found at reference [6].

Experiments/Results/Discussion

	Basic Random Forest	**Bayesian Optimization**	**Hyperparameter-optimized Bayesian Optimization**
Runtime (seconds)	0.24361324310302	33.5937597751617	20.9699952602386
Accuracy	0.7414	0.7586	0.7586
Precision	0.7692	0.8333	0.8333
Recall	0.4545	0.4545	0.4545

Table 1: Experimental Results

		Actually True	**Actually False**
Basic Random Forest	**Predicted True**	33	3
	Predicted False	12	10

Bayesian Optimization	**Predicted True**	34	2
	Predicted False	12	10
Hyperparameter-optimized Bayesian Optimization	**Predicted True**	34	2
	Predicted False	12	10

Table 2: Confusion Matrices

As shown in Table 1, the baseline model had an accuracy of 0.7414, precision of 0.7692, and recall of 0.4545, finishing execution in 0.24 seconds. Bayesian optimization on the second configuration increased accuracy to 0.7586 and precision to 0.8333 but maintained the same recall of 0.4545. This comes at a high computational cost; training took 33.59 seconds. The third experiment optimized the Bayesian optimization process, maintaining the same accuracy and precision but reducing execution time to 20.97 seconds, hence showing a more efficient tuning process. Note that this runtime is the time it took for the final best Bayesian Optimization process to run, not for the total search to find the best Bayesian hyperparameters. It is inefficient to run our search on your entire dataset, but better to run it on a smaller subset to determine good hyperparameters for a Bayesian Optimization run on the whole dataset.

From the confusion matrices shown in Table 2, in all three models, 12 cases of recurrence events were wrongly classified, which hints at the inability of the models to identify the positive label instances. While Bayesian optimization managed to provide better precision, reducing the count of false positives, it did not improve recall, meaning it failed in finding actual recurrence cases. The third experiment execution time reduced shows that we determined good hyperparameters for Bayesian optimization that could achieve just as good results in faster time than ones you might come up with yourself. If we obtained more entries for this dataset, we would be able to use the hyperparameters we found to be the best instead of running Bayesian optimization with self-engineered ones. From this perspective, our algorithm showed improvement and positive results,

but it would have been even more promising if there was an improvement in accuracy, precision, or recall.

Conclusion/Future Work

This study examined the impact of meta-optimizing Bayesian Optimization for hyperparameter tuning in Random Forest classifiers. By allowing Bayesian Optimization to fine-tune its own hyperparameters, we aimed to improve classification accuracy and efficiency compared to manually selected parameters and standard Bayesian Optimization. Our results showed that while Bayesian Optimization improved accuracy and precision over a baseline Random Forest, it came at a significantly higher computation cost. However, our meta-optimized Bayesian Optimization reduced execution time while maintaining the same classification performance. This demonstrates our model's potential for making hyperparameter tuning more efficient. Despite these improvements, the Bayesian Optimization and meta-based Bayesian Optimization models struggled to improve recall.

Looking ahead, there are several ways to expand on this work. First, testing with larger and more diverse datasets would provide insight into our model's generalizability to different fields. Second, we will evaluate our model's performance on other machine learning models, such as Gradient Boosting Machines (GBM) or Support Vector Machines (SVM). Third, we could have explored alternate surrogate models, such as Tree-structured Parzen Estimators (TPE) or neural networks. Finally, we could have investigated more ways to improve recall, perhaps through coding different feature selection or ensemble methods, for more reliable classification.

Contributions

Abstract - Andrew
Introduction - Gabriel
Related Work - Andrew
Dataset and Features - Gabriel
Methods - Gabriel
Experiments/Results/Discussion - Gabriel
Conclusion/Future Work - Andrew
References/Bibliography - Andrew

Bibliography

[1] Garrido-Merchán, E. C., & Jariego-Pérez, L. C. (2021). *Towards automatic Bayesian optimization: A first step involving acquisition functions*. **Lecture Notes in Computer Science, 12882**, 160–169. https://doi.org/10.1007/978-3-030-85713-4_16

[2] Lindauer, M., Feurer, M., Eggensperger, K., Biedenkapp, A., & Hutter, F. (2019). *Towards assessing the impact of Bayesian optimization's own hyperparameters*. **Proceedings of the DSO Workshop at IJCAI 2019**. arXiv:1908.06674. https://doi.org/10.48550/arXiv.1908.06674

[3] Snoek, J., Larochelle, H., & Adams, R. P. (2012). *Practical Bayesian optimization of machine learning algorithms*. **Advances in Neural Information Processing Systems, 25**, 2951–2959. https://proceedings.neurips.cc/paper/2012/file/05311655a15b75fab86956663e1819cd-Paper.pdf

[4] Thornton, C., Hutter, F., Hoos, H. H., & Leyton-Brown, K. (2013). *Auto-WEKA: Combined selection and hyperparameter optimization of classification algorithms*. **Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining**, 847–855. https://doi.org/10.1145/2487575.2487629

[5] Zhang, Z., Li, M., Wang, Y., & Chen, W. (2019). *Hyperparameter optimization for machine learning models based on Bayesian optimization*. **Journal of Computer Science and Technology, 34**(3), 509–522. https://doi.org/10.1016/j.jcst.2019.03.002

[6] Xu, G. & Chen, A. (2025). *Time-Optimized Bayesian Optimization of Random Forest Hyperparamaters.*

https://colab.research.google.com/drive/1ClEoIvJTcNjGLrQscv-BAarumYDO5_pj?usp=sharing.

[7] Eibe Frank, Mark A. Hall, and Ian H. Witten (2016). The WEKA Workbench. Online Appendix for "Data Mining: Practical Machine Learning Tools and Techniques", Morgan Kaufmann, Fourth Edition, 2016.

[8] Zwitter, M. & Soklic, M. (1988). Breast Cancer [Dataset]. UCI Machine Learning Repository. https://doi.org/10.24432/C51P4M.